



US005991830A

United States Patent [19]

Beard et al.

[11] **Patent Number:** 5,991,830[45] **Date of Patent:** Nov. 23, 1999

[54] **APPARATUS AND METHOD FOR COUPLING MULTIPLE PERIPHERAL DEVICES TO A SINGLE PORT OF A COMPUTER**

[75] **Inventors:** Paul F. Beard, Milpitas; Mark D. Moore, Palo Alto, both of Calif.

[73] **Assignee:** Compaq Computer Corp.

[21] **Appl. No.:** 08/885,144

[22] **Filed:** Jun. 30, 1997

Related U.S. Application Data

[63] Continuation-in-part of application No. 08/583,206, Jan. 4, 1996, Pat. No. 5,828,899.

[51] **Int. Cl.⁶** G06F 13/00

[52] **U.S. Cl.** 710/18; 710/2; 710/8; 710/37; 710/101; 710/137

[58] **Field of Search** 395/828-80 L, 395/281, 284; 710/2-4, 8-12, 15-19, 36-47, 62-64, 72-74, 101-104, 129-131

References Cited**U.S. PATENT DOCUMENTS**

4,792,970 12/1998 Molnar 379/284
5,276,443 1/1994 Gates et al. 340/825.06

5,615,404 3/1997 Knoll et al. 395/882
5,781,798 7/1998 Beatty et al. 395/830
5,784,576 7/1998 Guthrie et al. 395/283
5,805,412 9/1998 Yanagisawa et al. 361/686
5,805,833 9/1998 Verdin 395/281
5,819,112 10/1998 Kusters 395/886
5,828,899 10/1998 Richard et al. 335/828
5,832,244 11/1998 Jolley et al. 395/309
5,841,424 11/1998 Kikinis 345/168

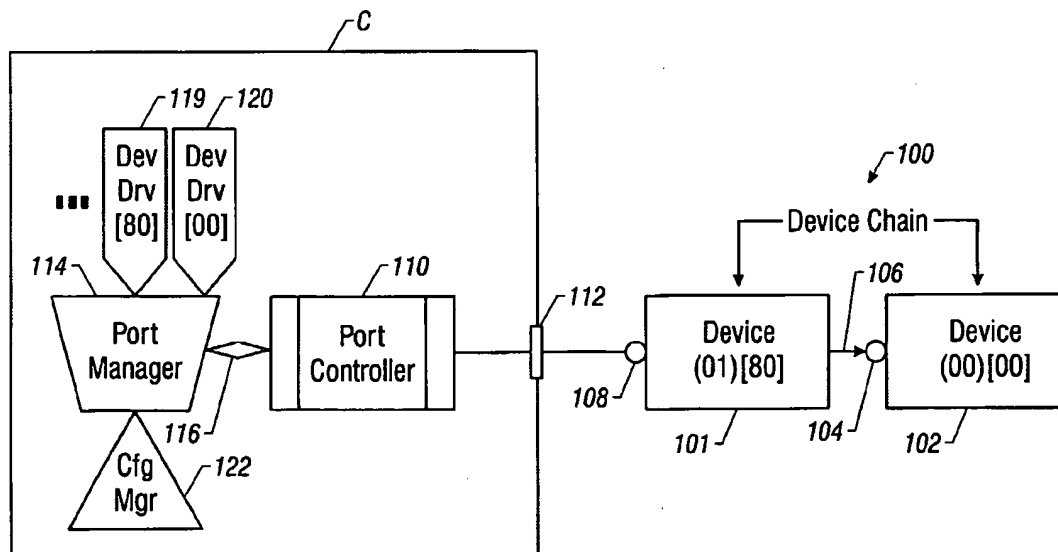
Primary Examiner—Christopher B. Shin

Attorney, Agent, or Firm—Ronald Chichester; Paul N. Katz, Frohwitter

[57]

ABSTRACT

A system for allowing a peripheral device to be inserted directly into a port of a computer system while the computer system is powered on. The insertion of a peripheral device into the computer system port is automatically detected, and a configuration operation is automatically performed when insertion of the peripheral device is detected. The system also allows a plurality of peripheral devices to be connected to a single port of a computer system by automatically determining the number of peripheral devices and assigning a unique address to each of the peripheral devices. The peripheral device may have a host port for communicating with the computer system, a slave port for connecting to a slave device, and a device manager which identifies if a slave device is connected.

20 Claims, 18 Drawing Sheets

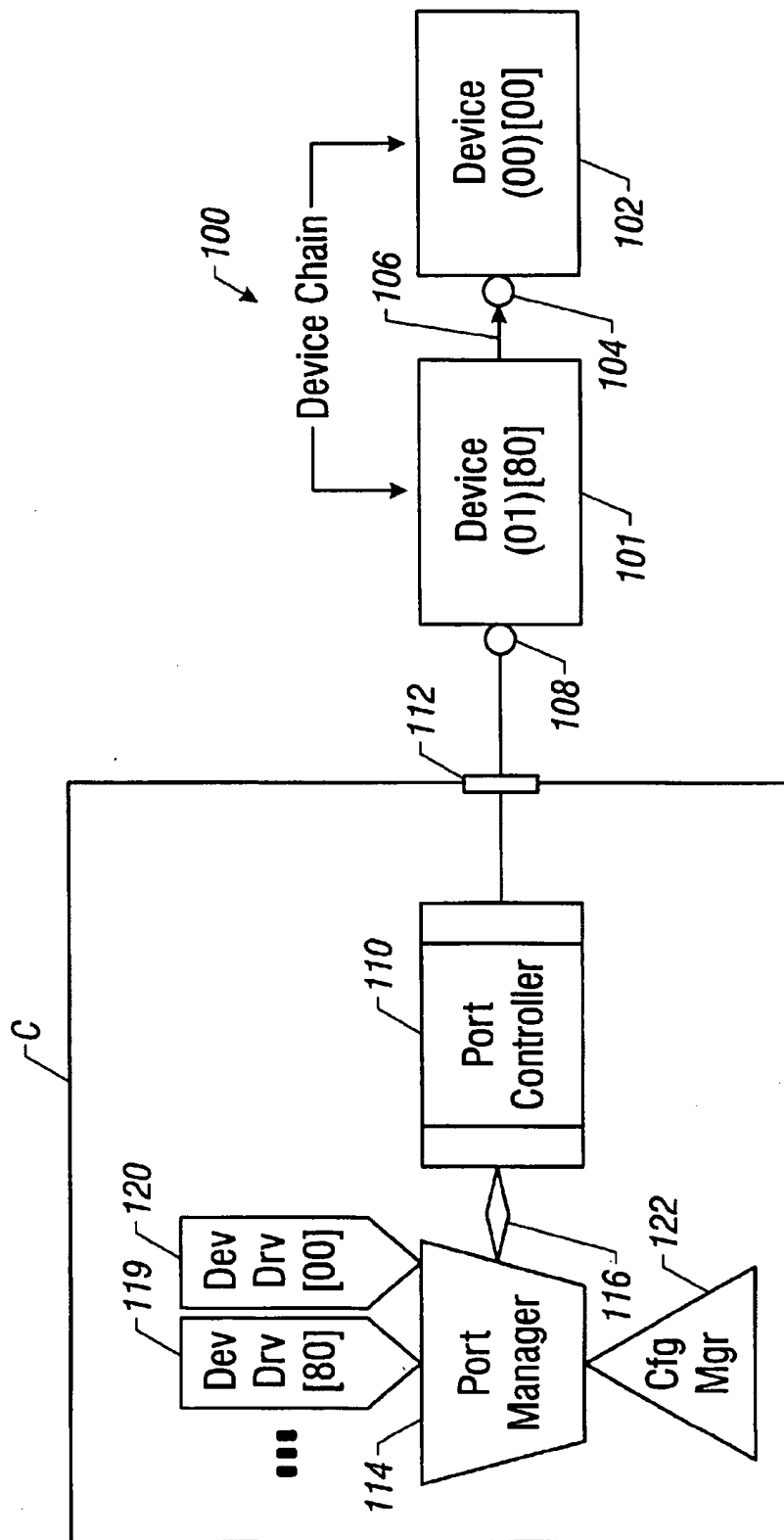


FIG. 1

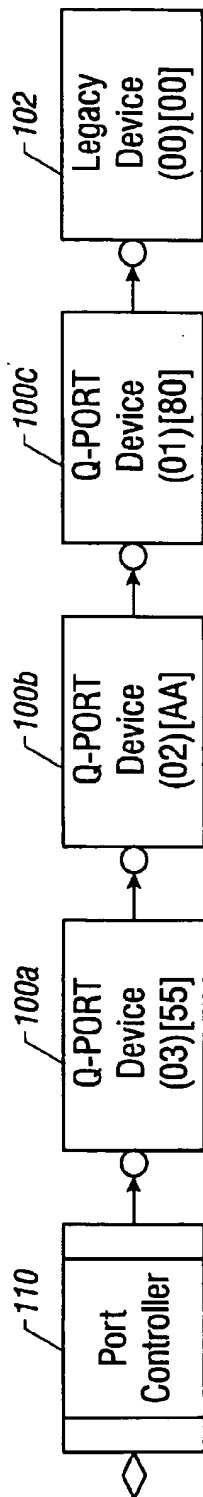


FIG. 2

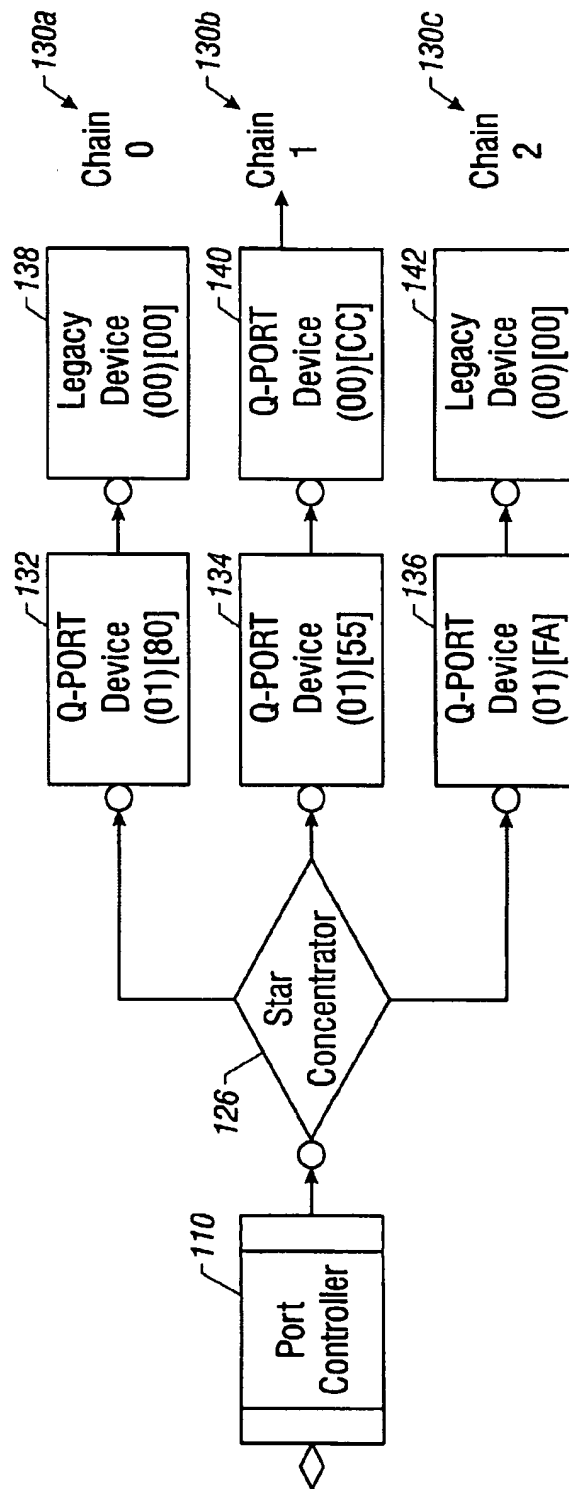
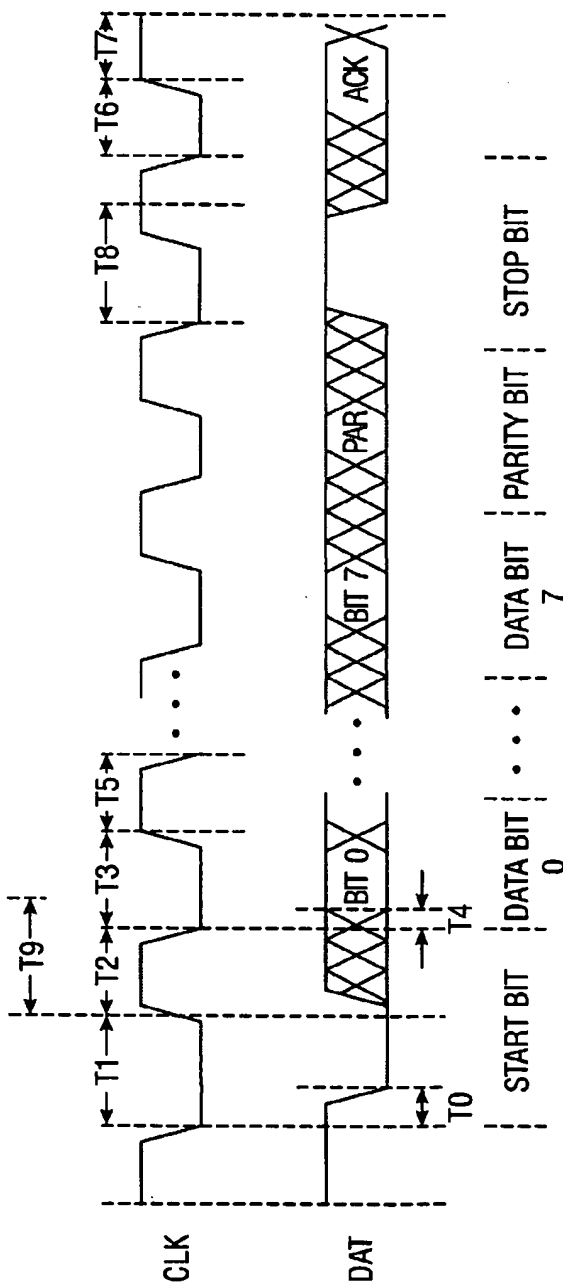


FIG. 3

FIG. 4

BYTE	7	6	5	4	3	2	1	0
0	Logic Device Address							
1	Reserved (0000)				Packet Length			
2	Data							
.
.
.
n	Data							

FIG. 5



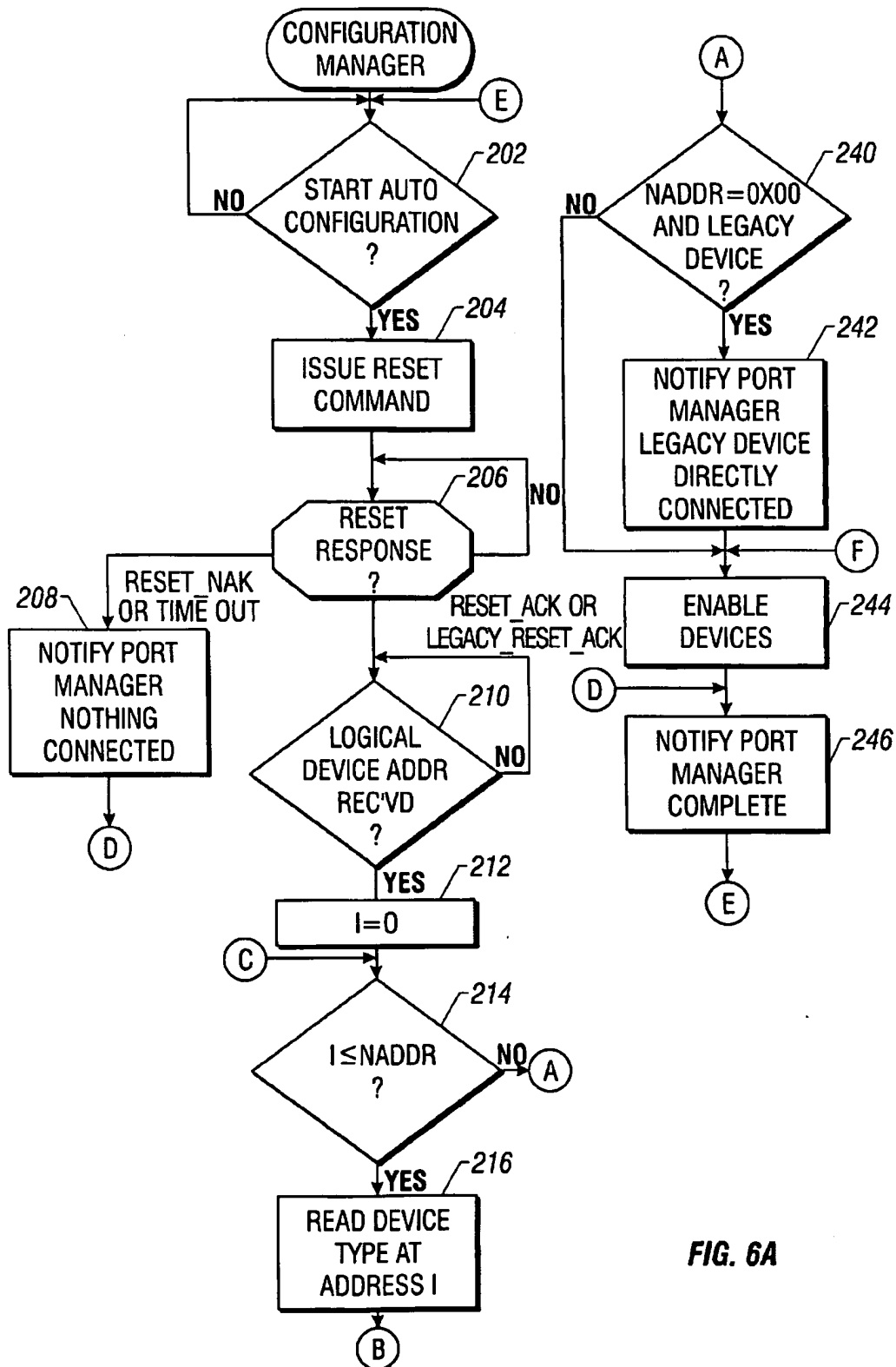


FIG. 6A

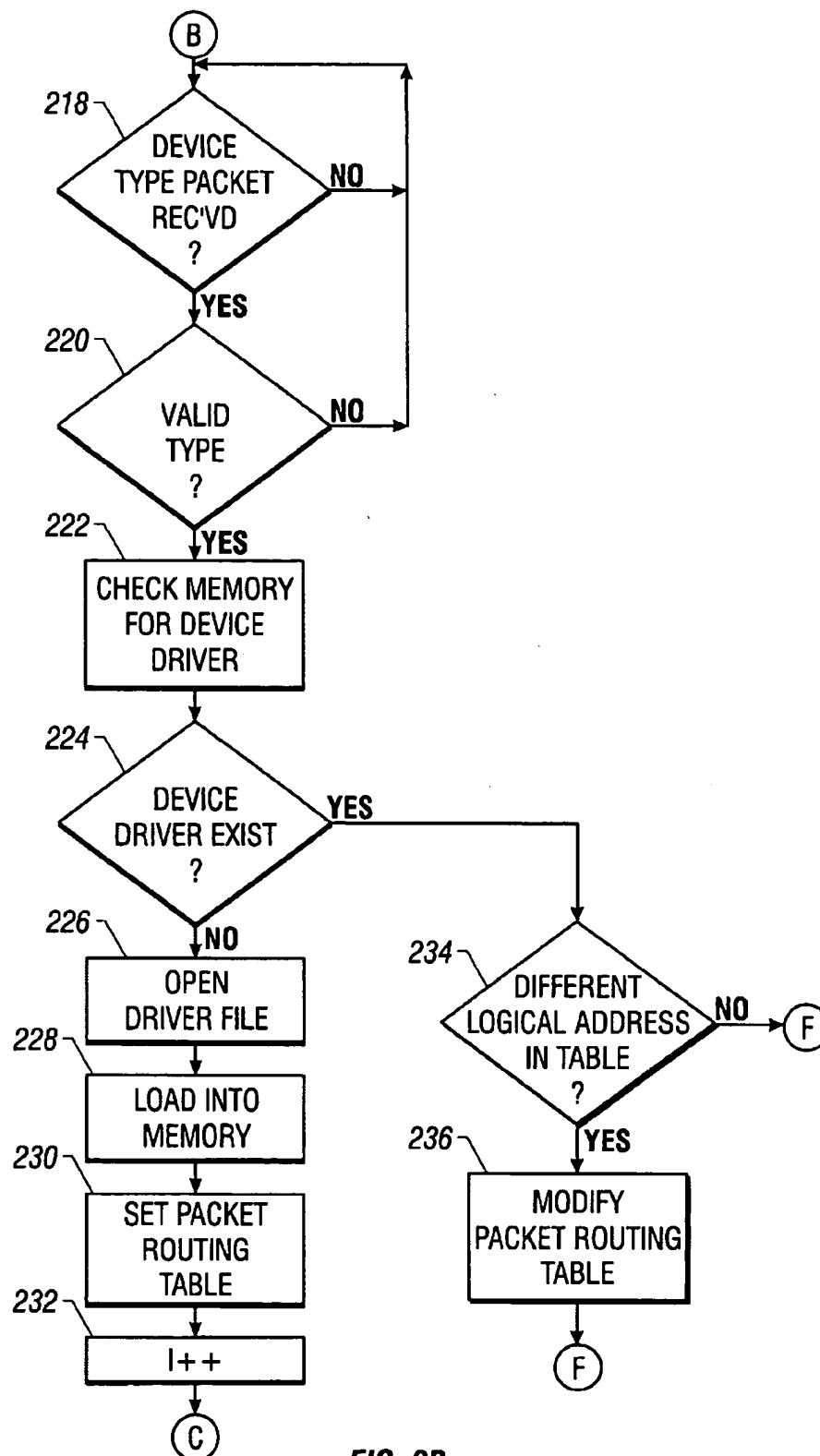


FIG. 6B

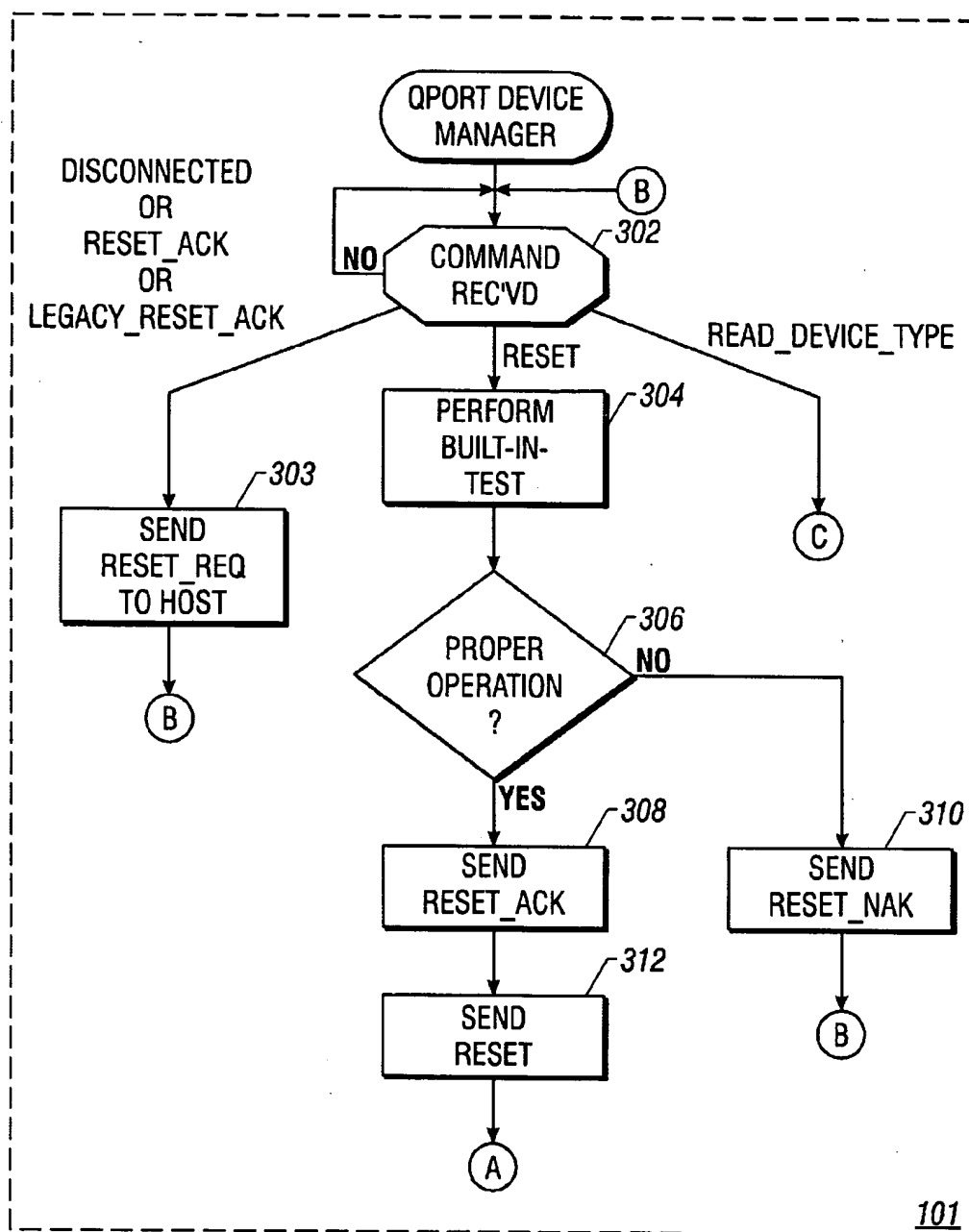


FIG. 7A

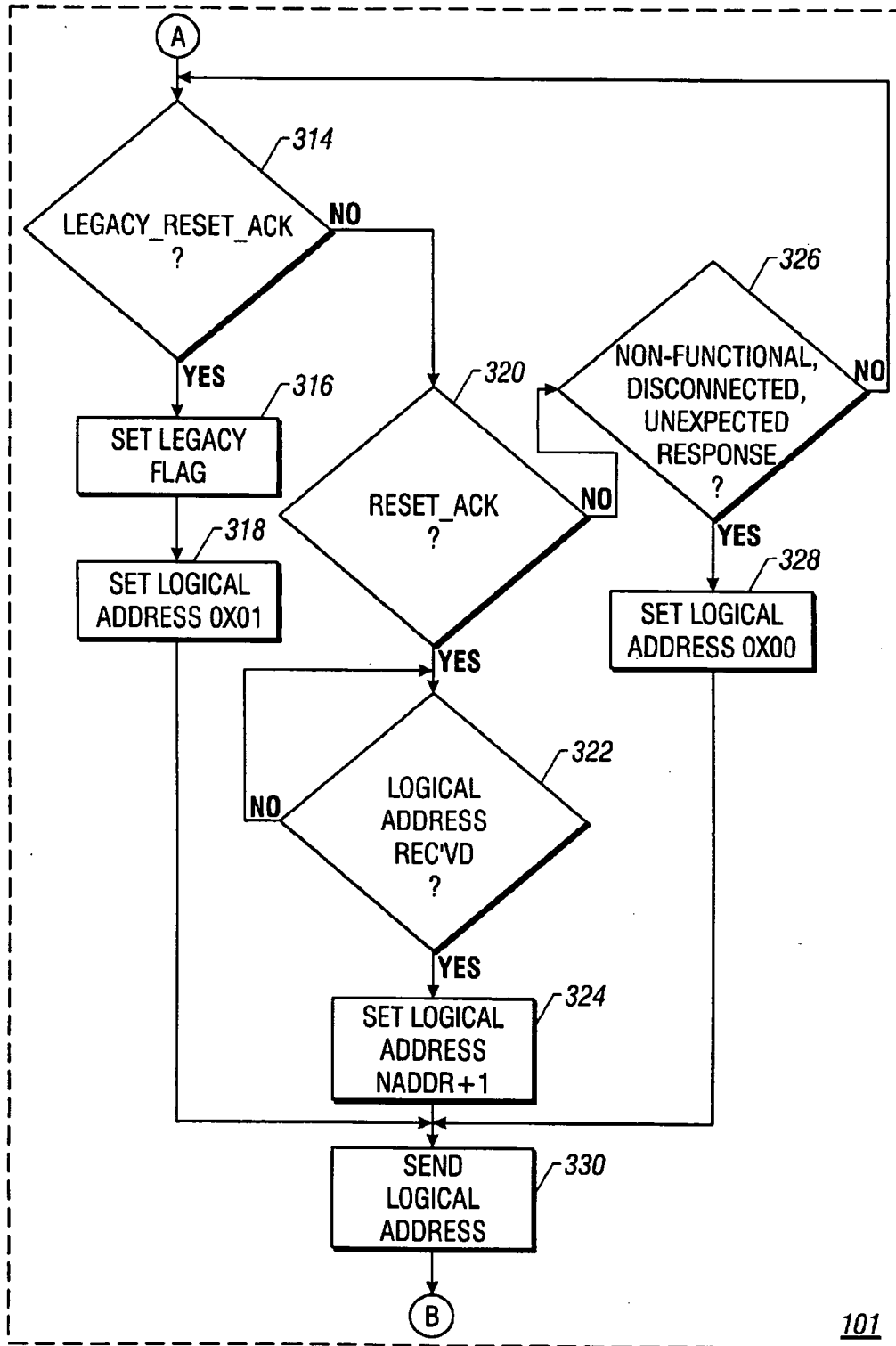


FIG. 7B

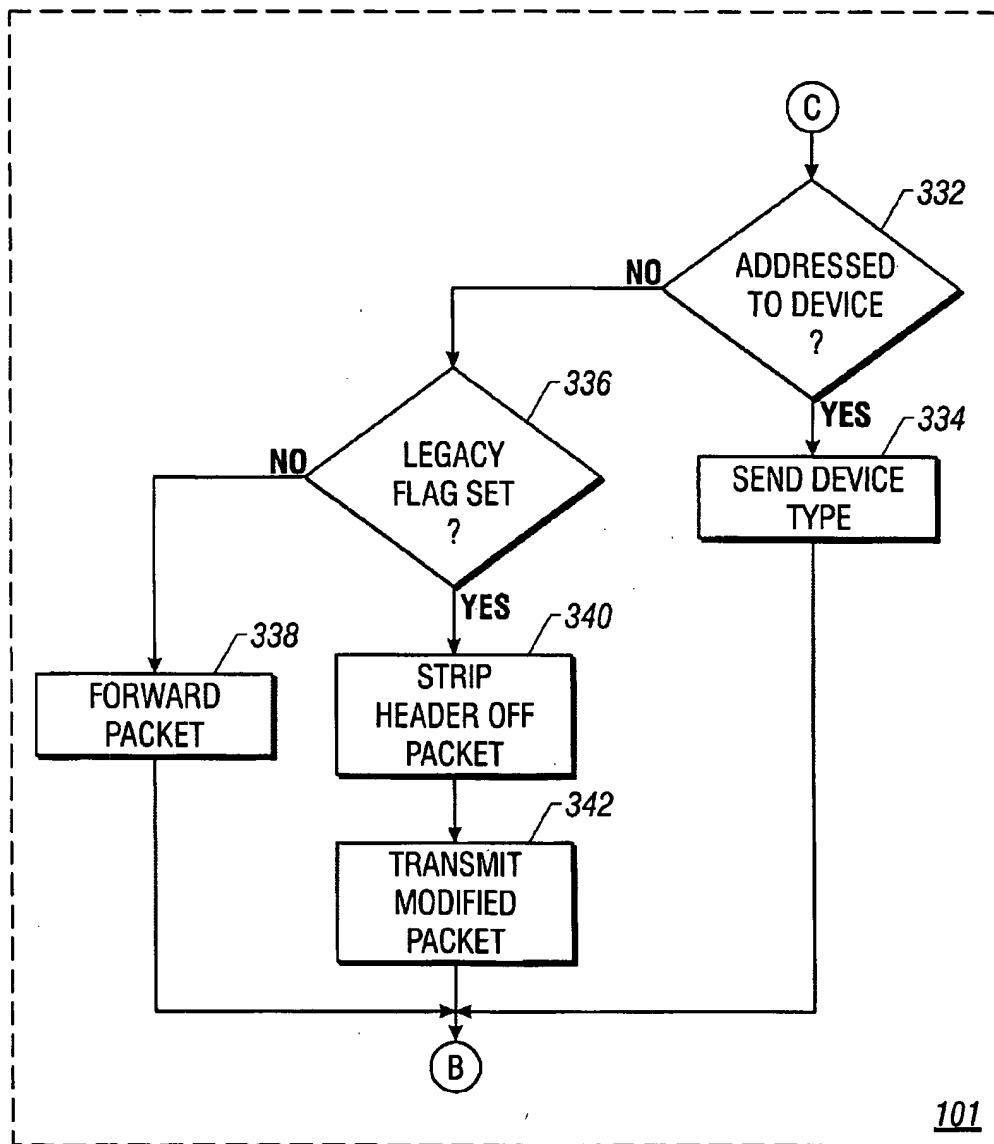


FIG. 7C

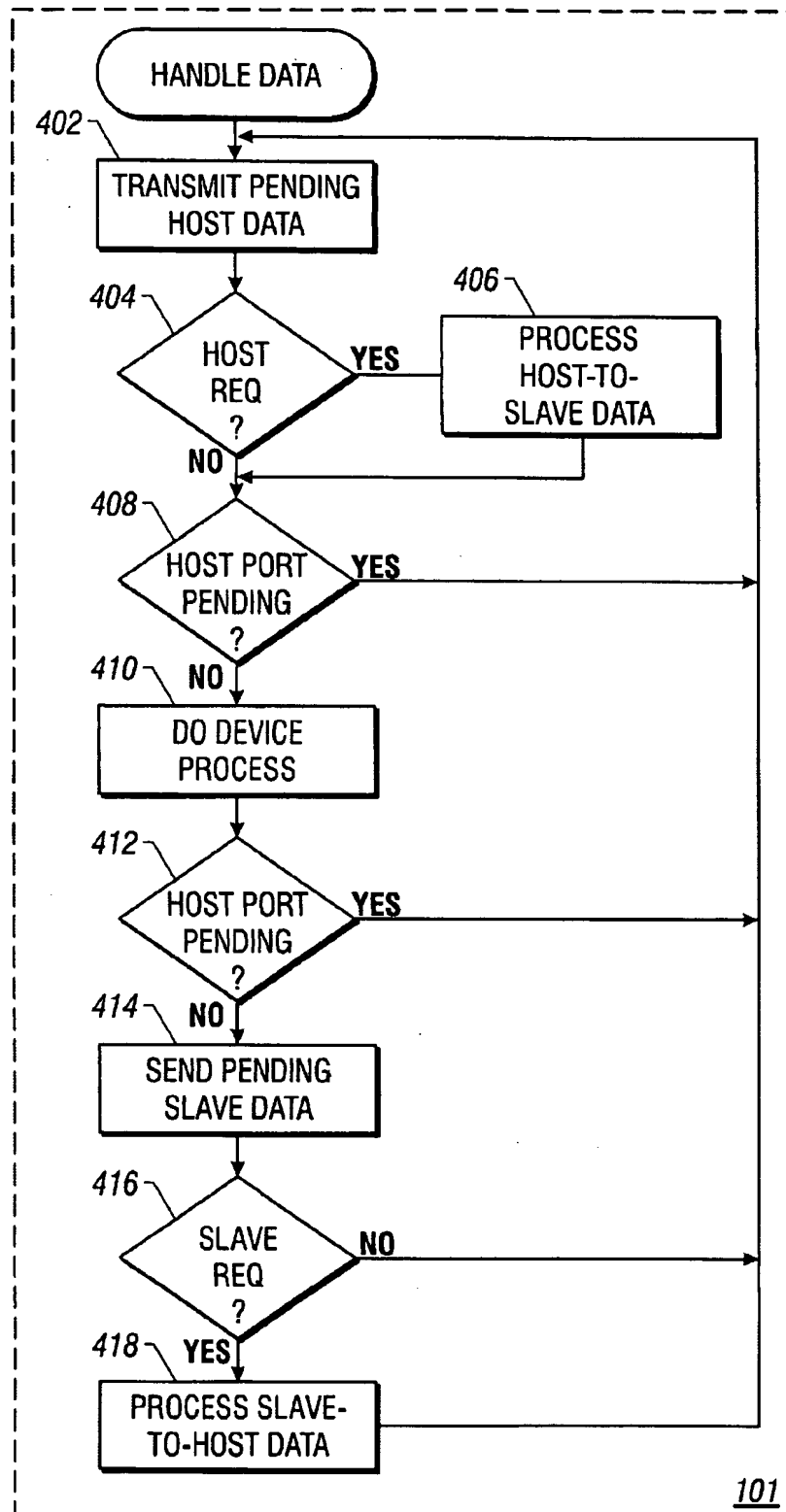
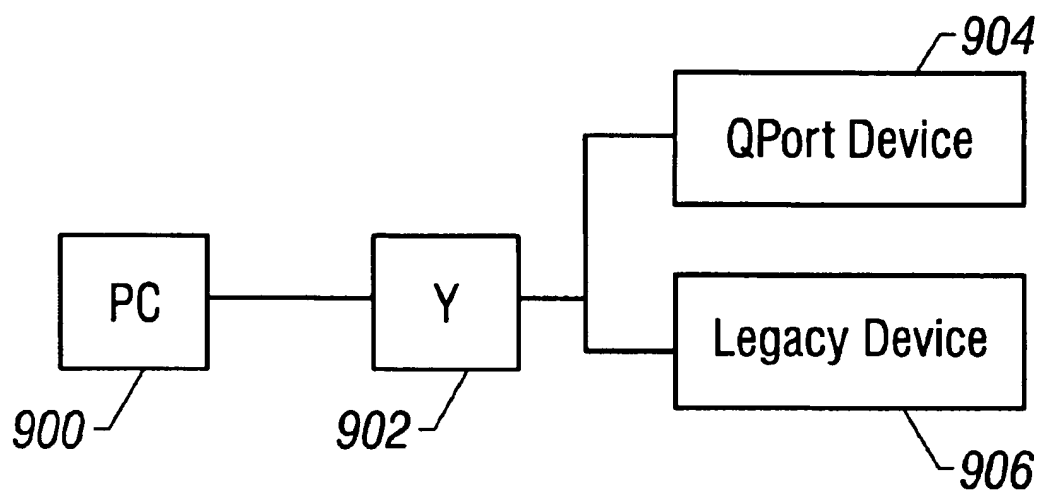


FIG. 8

**FIG. 9**

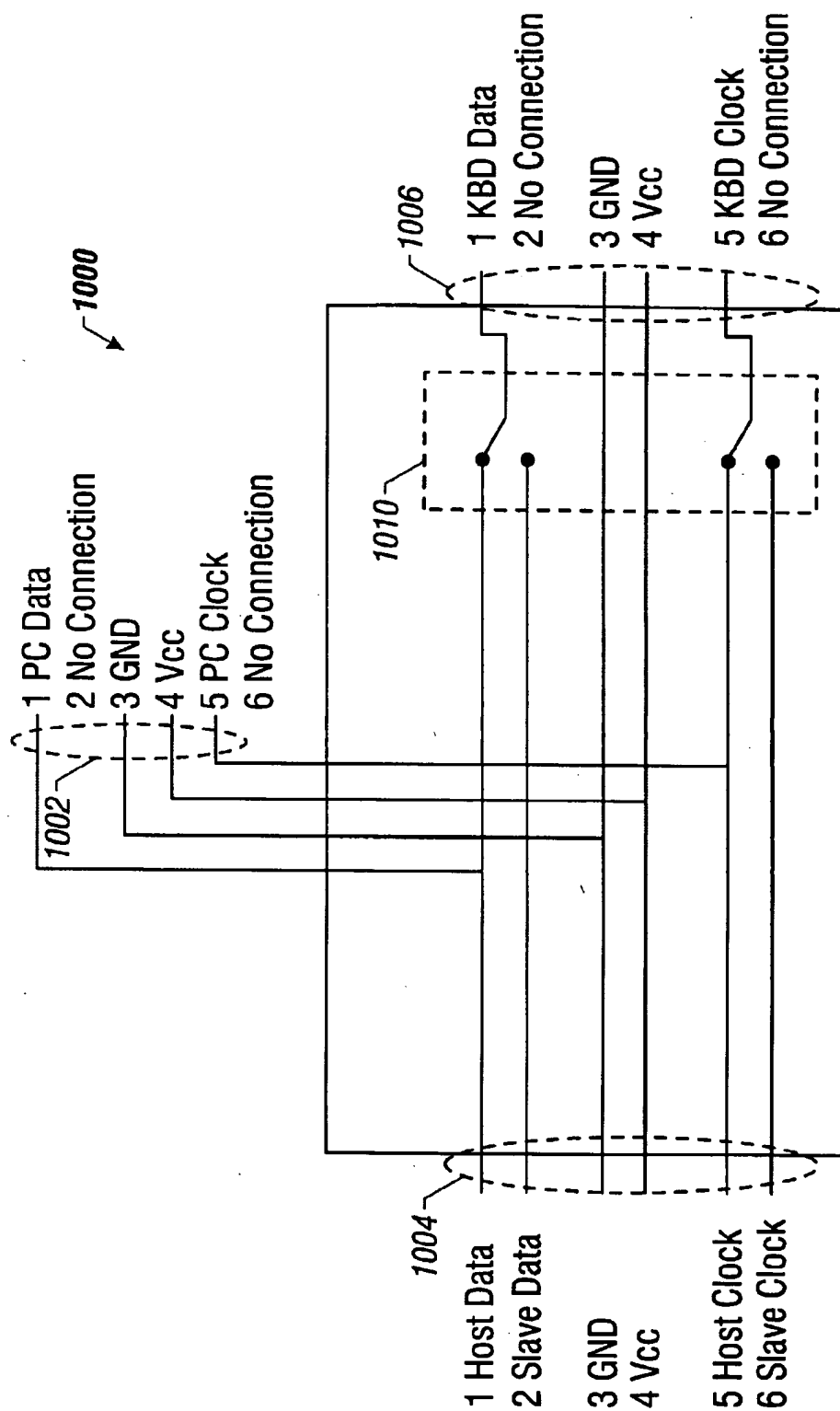


FIG. 10A

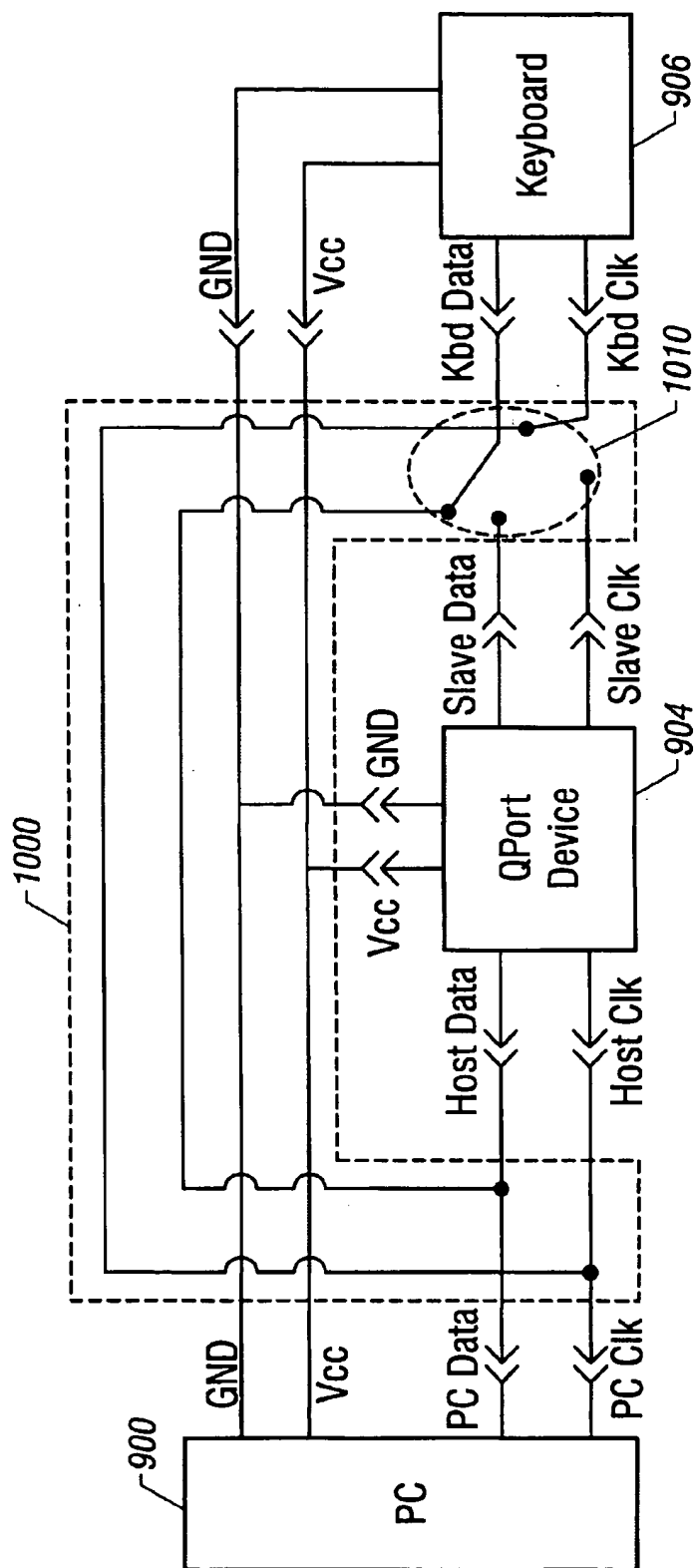


FIG. 10B

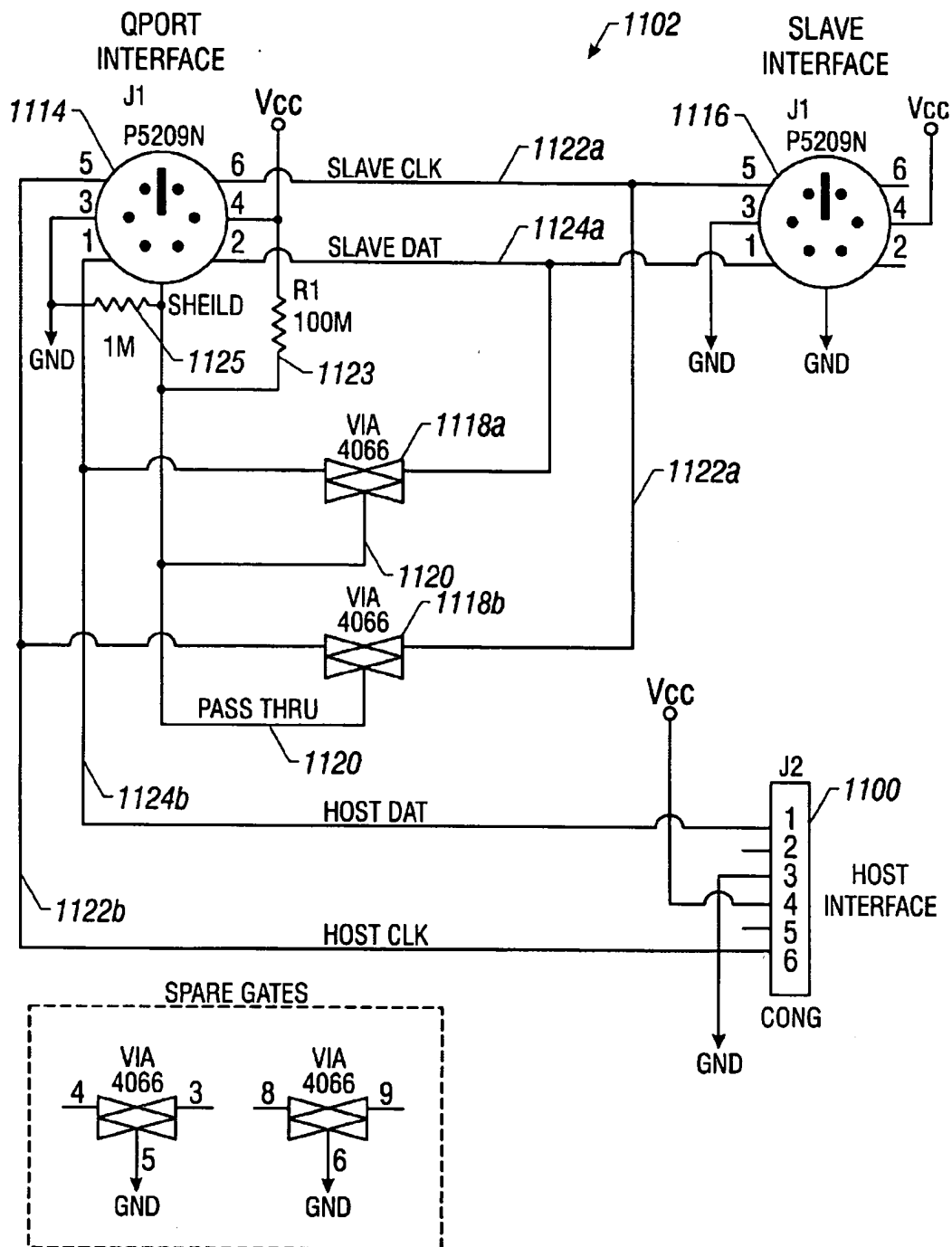
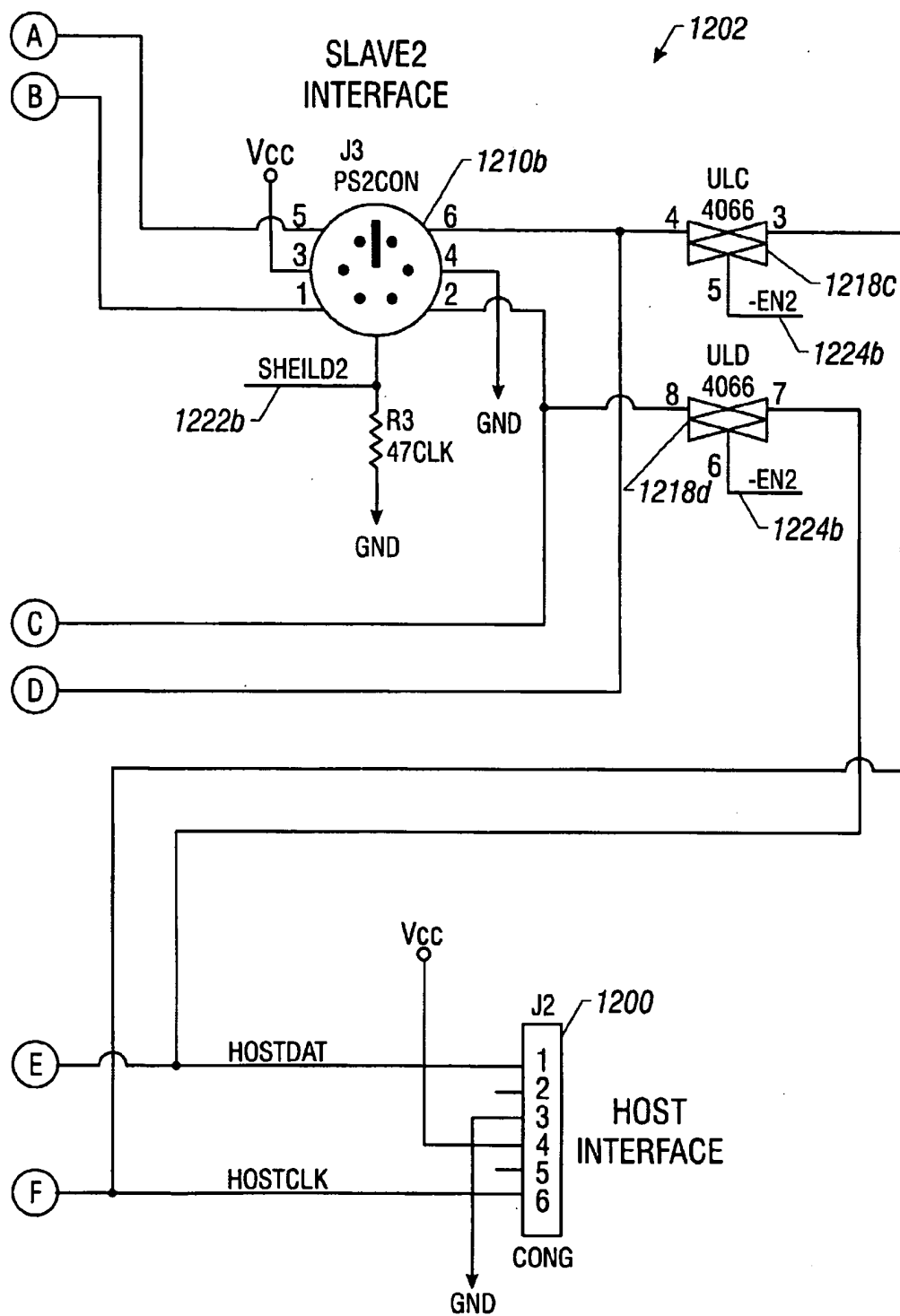


FIG. 11

**FIG. 12B**

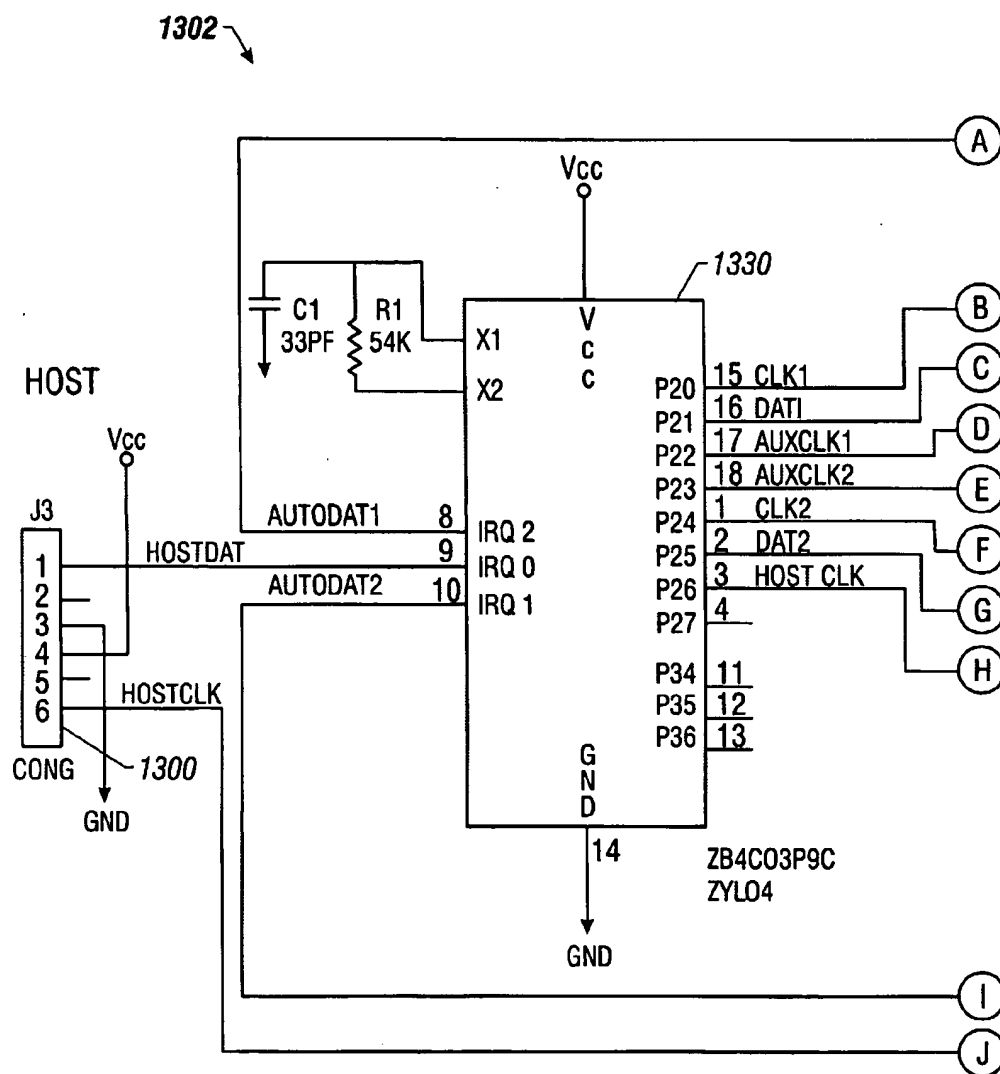


FIG. 13A

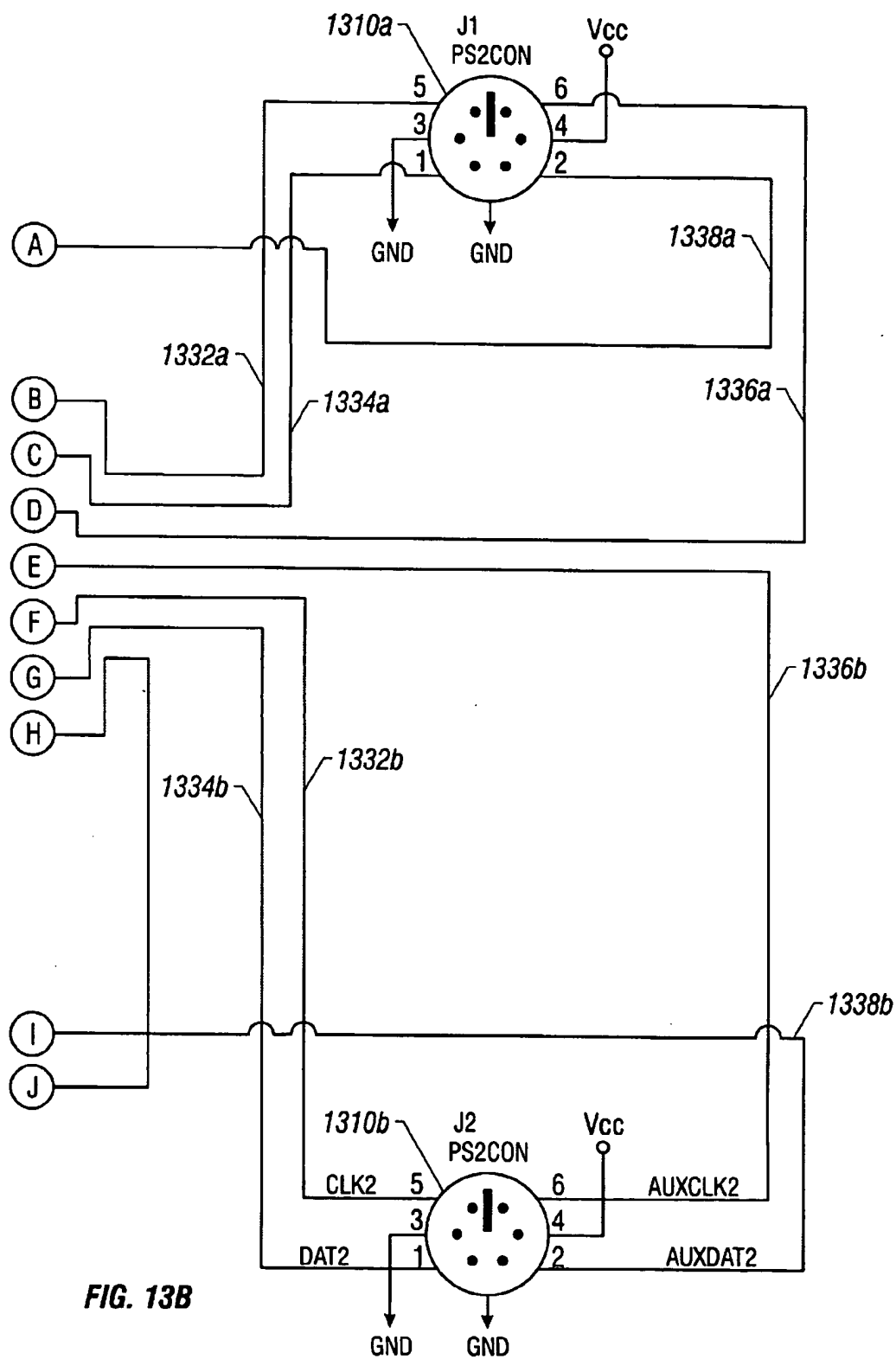
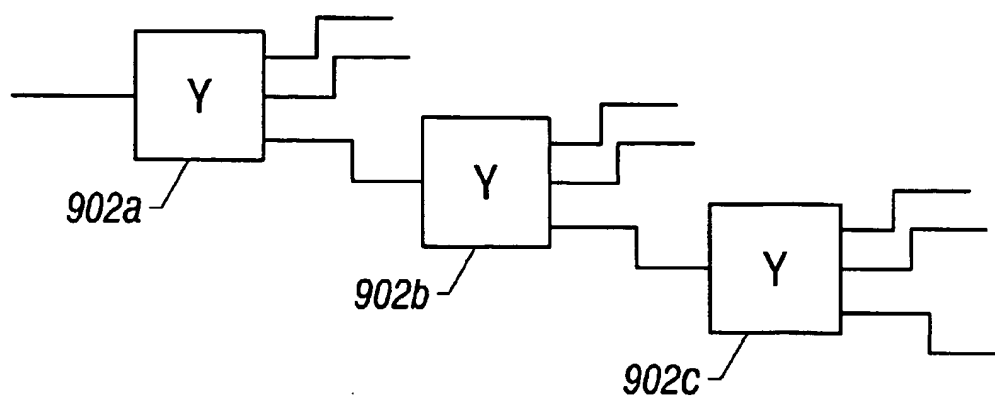
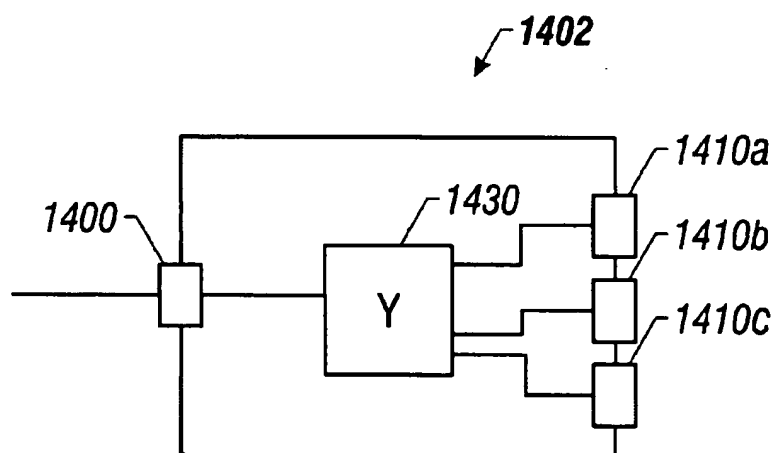


FIG. 13B

**FIG. 14****FIG. 15**

APPARATUS AND METHOD FOR COUPLING MULTIPLE PERIPHERAL DEVICES TO A SINGLE PORT OF A COMPUTER

This application is a continuation-in-part of a previously filed application entitled "Insertion of Peripheral Devices in Computer Systems", Ser. No. 08/583,206, filed on Jan. 4, 1996, now U.S. Pat. No. 5,828,899 herein incorporated by reference.

BACKGROUND OF THE INVENTION

This invention relates to computer peripheral devices.

A peripheral port, such as a keyboard port, a mouse port, or a serial interface port, is used in a computer system to connect to a peripheral device. Generally, each peripheral port is designed for connection to a specific type of peripheral device and is connectable to only one peripheral device. Solutions exist in which customized hardware interfaces are attached to an existing computer peripheral port to allow different types of and multiple peripheral devices to be attached to the computer port. Installation of the customized hardware requires that the computer case be opened.

SUMMARY OF THE INVENTION

In general, in one aspect, the invention features a system for allowing a peripheral device to be inserted directly into a port of the computer system while the computer system is powered on. The insertion of a peripheral device into the computer system port is automatically detected, and a configuration operation is automatically performed when insertion of the peripheral device is detected.

Implementations of the invention include the following features. The configuration operation may also automatically be performed once removal of the peripheral device from the computer system port is detected. The computer system may include a storage device which contains a device driver, which is retrieved during the configuration operation. The computer system may also include a memory, which is loaded with the device driver during the configuration operation. In addition, the configuration operation may also check for a device type of the peripheral device, with the device driver retrieved from this first device based on the device type. The configuration operation may also include determining a logical address of the peripheral device. The second peripheral device can be connected to the first peripheral device. The configuration operation may include issuing a configuration command to the first peripheral device. If the first peripheral device receives the configuration command and determines that the second peripheral device is connected, the first peripheral device may forward the configuration command to the second peripheral device. The logical address for the first peripheral device may be set according to whether the second peripheral device is connected to the first peripheral device. In addition, the peripheral device may also perform a self test during the configuration operation.

In general, in another aspect, the invention features a system for allowing a plurality of peripheral devices to be connected to a single port of a computer system. The number of peripheral devices connected to the single port is automatically determined, and an unique address is assigned to each of the peripheral devices.

Implementations of the invention include the following features. Data may be routed between the computer system and the peripheral devices based on an address associated

with the data. The type of each peripheral device may also be automatically determined. The computer system may include a memory, and the plurality of device drivers may be automatically loaded into the computer system memory with each device driver corresponding to a peripheral device. A routing table may also be set up having a plurality of entries, with each entry corresponding to a peripheral device and containing a device driver name and the address of the corresponding peripheral device. Data may be routed between the device drivers and the peripheral devices based on the routing table and an address associated with the data. Entries may be added to the routing table as additional peripheral devices are added, and entries may be removed from the routing table as peripheral devices are removed. Peripheral devices may be connected in one of the plurality of hardware configurations. A first peripheral device may be connected directly to the computer system port, with at least one other peripheral device being connected to the computer system port through the first peripheral device. The computer system port may be a serial port.

In general, according to another aspect, the inventions features a peripheral device for use with a computer system. The peripheral device has a host port for communicating with the computer system, a slave port for connecting to a slave device, and a device manager which identifies if a slave device is connection.

Implementations of the invention include the following features. The device manager may also route data between the computer system, the peripheral device, and the slave device. Automatic address assignment may be performed by the device manager based on whether the slave device is connected. The device manager may perform a power on sequence to determine if it is functioning properly. The device manager may also notify the computer system if the slave device is added to or removed from the slave port.

In general, according to another aspect, the invention features a peripheral device for use with a computer system having a peripheral port. The peripheral device includes a host port for directly connecting to the computer system peripheral port, a user interface connected to the host port, a slave port connected to the host port, and a device manager which notifies the computer system when it is inserted into the computer system peripheral port.

In general, according to another aspect, the invention features a port manager in a computer system having a device driver, a configuration manager, and a peripheral port for connecting to a peripheral device. The port manager includes a first device interfaced to the peripheral port for communicating with the peripheral device, and a second device interfaced to the device driver and the configuration manager for routing data between the device driver and the peripheral device and for routing configuration information between the configuration manager and the peripheral device.

In general, according to another aspect, the invention features a configuration manager in a computer system having a port manager and a peripheral port for connecting directly to a peripheral device. The configuration manager includes a first device interfaced to the port manager for receiving the notification. The configuration also includes a second device which responds to the notification by performing a configuration operation.

Implementations of the invention have one or more of the following advantages. Different types of peripheral devices may be connected to a single existing port. Multiple peripheral devices which may be arranged in a plurality of different

configurations may be connected to the existing port. The peripheral devices may be inserted into or removed from the port while the computer system remains powered on. No additional hardware need be added to the computer system to support the features listed above.

Other advantages and features will become apparent from the following description and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic block diagram of a peripheral device attachment scheme.

FIG. 2 is a schematic block diagram of a serial chain of peripheral devices.

FIG. 3 is a schematic block diagram of peripheral devices attached in a "star" arrangement.

FIG. 4 is a block diagram of a data packet.

FIG. 5 is a timing diagram of a data transfer.

FIGS. 6A and 6B are a flow diagram of an configuration manager for performing auto-configuration cycles.

FIGS. 7A, 7B and 7C are a flow diagram of a device manager for controlling a peripheral device.

FIG. 8 is a flow diagram of the device manager of FIGS. 7A, 7B and 7C for handling data transfer requests.

FIG. 9 is a schematic block diagram of a peripheral device attachment scheme including a peripheral router.

FIGS. 10a and 10b are schematic diagrams of a peripheral router.

FIG. 11 is a schematic diagram of another peripheral router.

FIG. 12 is a schematic diagram of another peripheral router.

FIG. 13 is a schematic diagram of another peripheral router.

FIG. 14 is a schematic diagram of a chain of peripheral routers.

FIG. 15 is a schematic diagram of another peripheral router.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 1, a so-called Q-PORT system includes the following layers: a device chain 100, a port controller 110, a port manager 114, a configuration manager 122, and device drivers 118 and 120. All the layers are contained in a host computer system C except the device chain 100. The port controller 110 (e.g., an 8042 microcontroller from Intel Corporation) is the hardware interface in the Q-PORT system and its associated software. The port controller 110 converts the serial bitstream used by the device chain 100 into a byte format used by software in the host computer system C. The port manager 114, which interfaces to the port controller 110 using a byte-oriented, interrupt driven software interface, is a low level software program which routes packetized data to and from the appropriate device driver. The port controller 110 signals the port manager 114 when a byte has been received from the device chain 100 and the port manager 114 can write a byte into the port controller 110 for distribution down the device chain 100. The configuration manager 122, which can be considered a special-purpose device driver for the Q-PORT system S, is a software program which keeps track of the hardware configuration of the Q-PORT system S and searches for and attaches the appropriate device drivers for the peripheral devices 101 and 102.

The device drivers 118 and 120, which communicate with the port manager 114 using a packet-oriented, interrupt driven software interface, act as the interface for data transferred between the device chain 100 and programs in the host computer system C, such as applications programs. With the exception of the configuration manager's interface, the interfaces to the device drivers 118 and 120 are standard. The port manager 114 signals a device driver when a packet destined for that device driver is received from the device chain 100. Header information is removed from the packet before it is transmitted to the device driver. The port manager 114 accepts a packet from a device driver and passes it on to the port controller 110, after header information is added to the packet, for transmission to the appropriate device in the device chain 100. The operations of the port manager 114, configuration manager 122, and port controller 110 are transparent to the device drivers 118 and 120.

The device chain 100 is made up of physical peripheral devices 101 and 102, which may be either legacy devices or Q-PORT devices that communicate serially. The legacy devices and Q-PORT devices can be standard peripheral devices such as a keyboard, a mouse or a joystick, or special-purpose peripheral devices such as game input devices (e.g. video game input pads), medical input devices (e.g. input control pads, sensors), or remote communication devices (e.g. infrared, microwave, or radio devices) for communicating with remote input devices. Each of the peripheral devices includes a user interface.

In FIG. 1, the peripheral device 102 is a legacy device and the peripheral device 101 is a Q-PORT device. The peripheral devices 101 and 102 can be added or removed and the Q-PORT system can reconfigure dynamically in an auto-configuration cycle. Each device is associated with a unique logical device address and device type set during the auto-configuration cycle. The legacy and Q-PORT devices can be connected by chaining, as shown by examples in FIGS. 1 and 2, or by attaching chains in a star configuration using a star concentrator, as shown by the example in FIG. 3.

The Q-PORT device 101 has a host port 108 connected to a computer port 112 and a slave port 106 connected to a host port 104 of the legacy device 102. The computer port 112 can be a standard port, such as a PS/2 style mouse port. Q-PORT and legacy devices can be inserted and removed from the computer port 112 while the computer system remains powered on, without requiring additional hardware interface.

The various ports may be connected using standard connectors such as 6-Pin Mini-DIN connectors. Each connector has a data or DAT signal and a clock or CLK signal. A device plugged into the slave port 106 is referred to as a "slave" or "down from" or "below" that Q-PORT device, and a device plugged into the host port 108 is referred to as a "host" or "up from" or "above" that Q-PORT device. The host computer system C is defined as being at the top of the chain. A Q-PORT device which has nothing connected to its slave port is at the bottom of the chain. In each of the configurations shown in FIGS. 1-3, only one computer port 112 is needed to connect to multiple peripheral devices. The functions performed by the Q-PORT device are controlled by a device manager in the Q-PORT device. The Q-PORT device is an "intelligent" peripheral device, and the device manager may be implemented with a microcontroller and associated software in the Q-PORT device.

Multiple Q-PORT devices each having a distinct logical address can be connected in a chain. The device at the

bottom of the chain has a logical device address of 0x00. Each device above increments the logical device address from the one below it (excluding reserved logical device addresses). For example, a chain with four devices is shown in FIG. 2, in which three Q-PORT devices 100A, 100B and 100C and the legacy device 102 are connected in series. Information issued from the top of the chain will either be acted upon by the device if it is addressed to that device, or it will be passed along to lower devices. Broadcast commands are both acted on as well as passed on. Information coming up the chain will be passed on without alteration (except in the case of a legacy device handler).

As shown in FIG. 3, in another example, three device chains 130A, 130B and 130C are connected together using a star concentrator 126, which appears as a host to the device chains 130A-C and which makes the chains appear as one long chain to its host. The star concentrator 126 translates the logical device addresses of the peripheral devices in the chains and passes data from the peripheral devices to the host computer system C.

The star concentrator 126 may optionally be incorporated in the port controller 110 and perform the logical device address translations after the serial-to-byte conversion, thus relieving the bandwidth requirements on the port controller's serial interface. The bandwidth would increase at the port controller's port-manager interface instead. With this option, the host computer system C sees multiple, functionally equivalent, slave ports and mechanical space, power supply limitations, and system bandwidth becomes the limiting factors on the fanout of the star connection rather than the bandwidth of the serial interface.

In the star configuration shown in FIG. 3, the star concentrator 126 performs the following logical device address translations for the port controller 110: address 0x00 of a legacy device 138 in device chain 130A is translated to address 0x00; address 0x00 of a legacy device 140 in the device chain 130C is translated to address 0x04; and so forth. In the example shown, the legacy device 138 and the legacy device 140 are of the same device type, e.g., device type 0x00. To allow proper routing by the port manager 114, each of the legacy devices 138 and 142 is associated with an individual device driver, as are Q-PORT devices 132, 134, 136 and 140.

Referring again to FIG. 1, the legacy device 102 has no slave port and is therefore, by definition, at the bottom end of the chain (logical device address of 0x00) as nothing can be plugged into it. The legacy device 102 can be connected directly to the computer system C or the last Q-PORT device in a device chain. The Q-PORT device 101 is capable of recognizing that the legacy device 102 is connected to its slave port 101 and supporting the legacy device 102 as a legacy device handler. Data going up the chain from the legacy device 102 is packetized by the legacy device handler and transported up the chain. Data coming down the chain has the header information removed by the legacy device handler before going to the legacy device 102.

Since a legacy device handler does not know the format and length of the packets of the legacy device 102, it may consider each byte received as a one-byte packet. Once the port manager 114 receives these packets, it passes the bytes on to the appropriate device driver individually. The device driver then re-assembles this stream of single bytes into a complete message.

Devices communicate with each other and with the port controller 110 through a serial bitstream directed through the host and slave ports. Each host port will have a male 6-Pin

Mini-DIN connector, and each slave port will have a female 6-Pin Mini-DIN connector. Both the host and slave ports are bi-directional. In addition, each port includes a CLK line and a DAT line, with each line connected to a pull-up resistor having a value of 2.2 k ohm, for example.

Host-to-slave transfers take priority over slave-to-host transfers. Q-PORT and legacy devices each has a buffer of sufficient size to hold an entire packet. Once the transfer of a packet begins, the entire packet must be transferred before any other information can be transferred from that port. Slave-to-host transfer bytes may interleave with host-to-slave bytes if the serial arbitration protocol allows for it, but all bytes in each packet must be received in each link without disturbing their order.

Referring to FIG. 5, a host-to-slave transfer is initiated by the host device first verifying that the serial interface is idle by checking that both the CLK and DAT lines are high. The host device then pulls the CLK line low and, within time period T0 (e.g. 5 μ s), it drives the start bit by pulling the DAT line low. The CLK line is held low for about time period T1 (e.g. 100 μ s) and then released which signals the slave device to clock the start bit in. The slave device then waits for about time period T2 (e.g. 40 μ s [+/-10 μ s]) with the CLK line released before the slave device pulls the CLK line low for about time period T3 (e.g. 40 μ s [+/-10 μ s]). After less than about time period T4 (e.g. 5 μ s), the host device places data bit 0 on the DAT line by pulling the DAT line low if a "011" and releasing the DAT line if a "11." When the slave device releases the CLK line again, it samples the DAT line. The slave device then leaves the CLK line released for about time period T5 (e.g. 40 μ s [+/-10 μ s]) before pulling it low to initiate the transfer of data bit 1. The process is repeated for data bits 0-7, an odd parity bit, and a stop bit (which is a 1). After the stop bit is transferred, the slave device pulls the CLK line low for about time period T6 (e.g. 40 μ s [+/-10 μ s]) and then the slave device places an acknowledge bit (0=ACK, 1=NAK) on the DAT line. The DAT line is released about time period T7 (e.g. 40 μ s [+/-10 μ s]) after the CLK line is released. If the host device samples a NAK acknowledge bit, which occurs if a parity error is detected, for example, then the host device can assume the transfer was unsuccessful and it can retry the transfer.

If a slave device does not cycle the CLK line within about time period T9 (e.g. 50 μ s) after the host device releases the CLK line in the start bit cycle, then a slave device is no longer connected to the host device's slave port. The host device detects this as the disconnected port state.

A slave-to-host transfer is performed in the same manner as the host-to-slave transfer, except that the slave device rather than the host device initiates the transfer by checking for an idle interface, the slave device drives the data bits, odd parity bit, and stop bit on the DAT line, and an acknowledge bit is not placed at the end. At the end of a slave-to-host transfer, if the host device needs extra time to process the received byte, it may inhibit or hold off further transfers by pulling the CLK line low within about a time period T8 (e.g. 50 μ s) after the slave device has pulled the CLK line low to begin the stop bit transfer. When the slave device releases the CLK line, it will notice that the CLK line remains driven, indicating that the transfer has been inhibited. The interface remains inhibited until the host device releases the CLK line, at which time the interface returns to the idle state.

If a parity error is detected in a slave-to-host transfer, then the host device may issue a REQUEST_RESEND command to the slave device for re-transmission. If, during a slave-to-host transfer, the host device finds that it is out of

synchronization with the slave device, the host device can signal a "framing error" condition by holding the DAT line low. This will be noticed by the slave device as soon as it attempts to drive a "1" on the DAT line, yet sees a "0" there instead. To regain synchronization, the slave device leaves the DAT line released and cycles the CLK line until it samples the DAT line pulled high again. The transfer then can be retried once the serial interface returns to the idle state.

While a host-slave interface is in the idle state, if either the host or slave device is unable to accept incoming data, or will be busy with another task long enough that it may miss the start bit, the device may inhibit the transfer by holding the CLK line low.

Communication in the Q-PORT system S takes the form of either broadcast communications or packet communications. Broadcast communications are single, double, or triple byte commands passed around the Q-PORT system S, such as during the logical device address assignment phase of the auto-configuration cycle. A packet moving down from the host computer system C is addressed to a specific device in a chain and a packet moving up from a device has the logical device address of the device which originated the packet.

Each packet includes header information followed by an optional data block. The header information contains a logical device address field to designate the target or source peripheral device and a packet descriptor field to specify the number of bytes in the data block. An example of a packet is shown in FIG. 4. The packet width is 8 bits and the data block can be from 0 to 16 bytes long, and the logical address field can be any 8-bit value except those reserved for broadcast commands.

Broadcast commands are used during the logical device address assignment phase of the auto-configuration cycle performed by the configuration manager 122. Broadcast commands are only used by the configuration manager 122 and are not be used by the device drivers 118 and 120. One broadcast command is the RESET command byte, which is issued by the configuration manager 122 in an auto-configuration cycle to cause every device in the device chain 100 to go to a known state and to recursively assign all the logical device addresses in the device chain.

Each of the following command bytes in the Q-PORT system S are transmitted as part of the data block in a packet. A SET_DEFAULT command is sent by the host computer system C to restore all settings in the peripheral device to its default value and to disable the device. A DISABLE command is sent to disable a peripheral device from sending any unsolicited packets while in periodic or event driven modes. The ENABLE command is sent to enable a device to send unsolicited packets while in periodic or event driven modes. After each of the above commands is received and executed, the device acknowledges by returning an ACK command in a packet to the host computer system C.

A READ_DEVICE_TYPE command causes the receiving device to respond with a packet containing the device's device type. The device type information is used during the auto-configuration cycle by the configuration manager 122. The host computer system C recognizes that this command was received and executed when it receives the device type packet.

The port manager 114 signals the configuration manager 122 to perform the auto-configuration cycle in response to several events: on initial entry of the port manager 114; when a RESET_REQ byte is received from some device; or by sending an unexpected RESET_ACK byte to the con-

figuration manager 122. During the auto-configuration cycle, the port manager 114 communicates only with the configuration manager 122 and ignores any other device drivers.

Referring to the flow diagram of FIGS. 6A-6B, the configuration manager 122 starts 202 an auto-configuration cycle when the port manager 114 informs it that the device chain configuration has changed. The configuration manager 122 first issues 204 a RESET command and waits 206 for a reset response. If a RESET_NAK byte is received, indicating a non-functional Q-PORT device, or a time-out is detected, the configuration manager 122 notifies 208 the port manager 114 that nothing is connected and also notifies 246 the port manager 114 that the auto-configuration cycle is complete. If the port controller code is not written to check for time-out conditions, a small piece of software may monitor the functions of the port controller 110 and check for time-outs. This software could report these errors to the port controller 110, which would pass these on to the port manager 114. The time-out checking function can also be included directly in the port controller code.

If the configuration manager 122 receives a RESET_ACK or LEGACY_RESET_ACK byte, it waits 210 for a logical device address NADDR to be transmitted from the device chain 100. A legacy device responds to a RESET command by issuing a 3-byte sequence: a LEGACY_RESET_ACK byte, a LEGACY_ACK byte, and a LEGACY_DEVICE_TYPE byte. The RESET_ACK, RESET_NAK, LEGACY_RESET_ACK, LEGACY_ACK, LEGACY_DEVICE_TYPE and logical device address bytes are transmitted as broadcast commands by the peripheral devices.

After a logical device address is received, the configuration manager 122 resets 212 an increment variable I to zero. If the configuration manager 122 determines 214 that the variable I is less than or equal to the value of the received logical device address NADDR, it transmits 216 a READ_DEVICE_TYPE command packet to the device at logical device address I. The configuration manager 122 then waits 218 for the receipt of a device type packet. During the auto-configuration cycle, the port manager 114 forwards the data block portion of any received packet to the configuration manager 122. After receipt of data, the configuration manager 122 determines 220 if the device type is a valid type. A legacy device responds to the READ_DEVICE_TYPE command by first issuing a LEGACY_ACK byte, which is an invalid device type. If an invalid type is detected 220, the configuration manager 122 waits 218 for another packet.

Once a valid device type packet is received, the configuration manager 122 checks 222 host computer system memory (not shown) to determine 224 if the device driver associated with the device type is loaded. If the device driver does not exist, the configuration manager 122 opens 226 the appropriate device driver file from the mass storage device, e.g., hard disk drive, of the host computer system C and loads 228 the device driver into host computer system memory. The configuration manager 122 then adds 230 an entry associated with the new device type into a packet routing table for use by the port manager 114. Each entry of the packet routing table includes the device driver name and the associated logical device address and device type of the physical peripheral device associated with the device driver. The configuration manager 122 then increments 232 the value of the variable I by one and checks 214 if the variable I is less than or equal to the received logical device address NADDR.

If the configuration manager 122 determines 224 that a device driver is already loaded in host computer system memory, it checks 234 the routing table to compare the received logical device address with the stored logical device address. If the addresses are different, then the configuration manager 122 modifies the appropriate routing table entry to associate the device driver to the correct logical device address.

If the configuration manager 122 determines 214 that the variable I has a value greater than the address NADDR, and if it determines 240 that the address NADDR has the value 0x00 and the connected peripheral device is a legacy device, the configuration manager 122 notifies 242 the port manager 114 that the legacy device is directly connected. A directly-connected legacy device does not have a Q-PORT device to act as its legacy device handler, which would require the port manager 114 to essentially act as a byte pipeline between the legacy device and the legacy device driver, performing no interpretation, other than handling broadcast commands.

The configuration manager 122 then sends 244 ENABLE command packets to each of the peripheral devices in the device chain to enable the peripheral devices. Next, the configuration manager 122 notifies 246 the port manager 114 that the auto-configuration cycle is complete.

The flow diagram of FIGS. 7A-C shows the response of the Q-PORT device to certain command bytes. If the Q-PORT device receives 302 a RESET command, it performs 304 a power-on reset cycle which includes a Built-In-Test (BIT) procedure. If the Q-PORT device determines 306 that it is functioning properly, it issues 308 a RESET_ACK byte to the host and issues 312 a RESET command to its slave. Any packets received by the slave port of the Q-PORT device must be flushed before the RESET command can be sent by the Q-PORT device.

The Q-PORT device expects the slave to respond in one of the following ways: a disconnected port state, indicating that the device is not connected to a slave; a RESET_NAK command, indicating a non-functional slave; a LEGACY_RESET_ACK command, indicating a legacy device is connected as the slave and has passed its BIT procedure; or a RESET_ACK command, indicating the slave is a Q-PORT device and has passed its BIT procedure.

If the Q-PORT device receives 314 the LEGACY_RESET_ACK command, then the Q-PORT device sets 316 its legacy internal flag to identify itself as the legacy device handler and sets 318 its logical device address at 0x01. If instead the Q-PORT device receives 320 a RESET_ACK command, indicating that the slave device is another Q-PORT device, it waits 322 for the slave device to send its logical device address. The Q-PORT device sets 324 its logical device address as the slave's logical device address plus one.

However, if the Q-PORT device discovers 326 that its slave port is disconnected, the slave is not functional, or an unexpected response is received, then the Q-PORT device assumes it is the last functional device in the device chain and sets 328 its logical device address at 0x00. After setting its logical device address, the Q-PORT device sends 330 the address to the host.

If the Q-PORT device receives 302 a reset response (RESET_ACK or LEGACY_RESET_ACK) from its slave without having first issued a RESET command to that slave, or it detects that the slave has become disconnected, then the Q-PORT device sends 303 a RESET_REQ byte (a broadcast command) up the device chain to the configuration manager 122, which responds by performing an auto-configuration cycle.

If the Q-PORT device receives 302 a READ_DEVICE_TYPE command and the command is addressed 332 to the Q-PORT device, the Q-PORT device sends 334 a packet containing its device type to the host. Normally, if the command packet is not addressed 332 to the Q-PORT device, it would simply forward 338 the packet to the slave, but if the internal legacy flag is set 336, the Q-PORT device strips 340 the header off the packet before passing 342 it on to the slave. The legacy device will respond to the READ_DEVICE_TYPE command by issuing a LEGACY_ACK byte followed by a LEGACY_DEVICE_TYPE byte.

Referring to the flow diagram of FIG. 8, the device manager in a Q-PORT device handles pending data transfer requests. The host port of the Q-PORT device contains a host FIFO queue for storing upstream data, and the slave port of the Q-PORT device contains a slave FIFO queue for storing downstream data. The host and slave FIFO queues each has the capacity to store at least one data packet. Initially, the device manager transmits 402 any upstream host data pending in the host FIFO queue. After a byte of data has been transmitted, if a host device transfer is pending 404 (i.e., host device has taken control of DAT line), priority is given to the host-to-slave transfer and the host device data transfer is processed 406.

If upstream data is pending 408 in the host FIFO queue, the device manager transmits 402 the next byte of the upstream data. However, if upstream data byte is not pending 408, any input from the user interface is processed 410. For example, if the Q-PORT device is a keyboard, the pressing of a keyboard by the user would be processed in this step. Next, if upstream data is pending 412 in the host FIFO queue, the data is transmitted 402. If upstream data is not pending 412, then all downstream data pending in the slave FIFO queue are transmitted 414. Next, if a data transfer request is pending 416 from a slave device, the slave-to-host data is transmitted 418.

The following provides an example of an auto-configuration cycle. To begin with, the host computer system C is unconnected with any peripheral devices. On power-up, the configuration manager 122 is initiated and issues a RESET command to the port manager 114. The port manager 114 passes this RESET command to the port controller 110, but as there are no devices present, the transfer times out and the port controller software responds with a RESET_REQ byte, which indicates to the configuration manager 122 that nothing is connected.

Next, a Q-PORT device (having a device type of 0x80) is inserted into the computer port 112. When the Q-PORT device receives power, it runs its power-on reset cycle, which (assuming BIT passed) sends a RESET_ACK byte back through the port controller 110. The port manager 114 recognizes the RESET_ACK byte as being a broadcast command and routes it to the configuration manager 122. The configuration manager 122 is not expecting receipt of the RESET_ACK byte, as the previous configuration cycle showed no devices were connected. In response, the configuration manager 122 issues a RESET command through the port manager 114 and port controller 110 to start an auto-configuration cycle.

If the Q-PORT device had already issued its logical device address, the port controller 110 ignores the RESET command. Otherwise, since the host transfer takes precedence, the RESET command is sent on to the Q-PORT device. Upon receipt of the RESET command, the Q-PORT device again does a BIT and responds with the RESET_ACK byte followed by a logical device address of 0x00. The port

manager 114 again recognizes these two bytes as being broadcast commands and routes them both to the configuration manager 122.

This time, the configuration manager 122 is expecting a RESET response of some sort, and so recognizes that there is one device in the device chain located at logical device address 0x00. The configuration manager 122 then issues a READ_DEVICE_TYPE command and the Q-PORT device returns a packet containing a device type of 0x80. The port manager 114 recognizes the incoming data as a packet, but realizes that the configuration manager 122 is still performing an auto-configuration cycle, so it routes the data block portion of the packet containing the device type to the configuration manager 122.

Next, the configuration manager 122 opens the device driver file in the mass storage device corresponding to a device type of 0x80. This device driver is loaded into memory, and the configuration manager 122 sets up a routing table for the port manager 114 to route packets having a logical device address of 0x00 to the loaded device driver, and conversely, that data from the device driver be packetized and sent to logical device address 0x00.

The configuration manager 122 then issues an ENABLE command to both the device driver and the port manager 114 to start the connection. Next, the configuration manager 122 exits from auto-configuration. Packets to and from the Q-PORT device are transferred according to the routing table until a hardware configuration change occurs.

Next, a legacy device (having device type 0x00) is connected to the slave port of the previously inserted Q-PORT device. When the Q-PORT device performed a power-on reset sequence after it was inserted, it determined that there was no device connected to its slave port. If a legacy device gets plugged into the slave port, the legacy device also runs a power-on reset sequence and a BIT. If the BIT passed, the legacy device issues to a LEGACY_RESET_ACK byte, a LEGACY_ACK byte, and a LEGACY_DEVICE_TYPE byte to its host, the Q-PORT device. As the Q-PORT device is not expecting receipt of the bytes, it realizes that its slave port connection has changed. As a result, it sends a RESET_REQ byte to its host port.

When the port manager 114 receives the RESET_REQ byte, it recognizes it as a broadcast command and routes it to the configuration manager 122. In response, the configuration manager 122 issues a RESET command to the port manager 114, which forwards this through the port controller to the device chain. The Q-PORT device receives this RESET command and performs a BIT. If successful, the Q-PORT device sends a RESET_ACK byte to its host port. Next, it sends the RESET command to its slave port. In response to the RESET command, the legacy device runs another BIT and resends the sequence containing the LEGACY_RESET_ACK byte, LEGACY_ACK byte, and LEGACY_DEVICE_TYPE byte.

The Q-PORT device sees this sequence and recognizes that its slave device is a legacy device because of the LEGACY_RESET_ACK byte, so the Q-PORT device sets an internal legacy flag to set itself as the legacy handler for its slave. At this point, the Q-PORT device knows that its slave device has a logical device address of 0x00, and it, therefore, must have a logical device address of 0x01. The Q-PORT device then issues the logical device address of 0x01 to its host port.

The port manager 114 had previously received the Q-PORT device's RESET_ACK byte, and so it was expecting the logical device address next. Both the RESET_ACK

byte and the logical device address are passed on to the configuration manager 122. The configuration manager 122 now knows there are two devices in the device chain. It first issues a packet containing the READ_DEVICE_TYPE command to the device at logical device address 0x00, which is received by the Q-PORT device. Normally, since the packet is not addressed to the Q-PORT device, it would simply pass it along to its slave port. However, since the Q-PORT device is the legacy device handler, and the logical device address on the packet is 0x00, the Q-PORT device strips the header off the packet and passes it on to its slave port.

Upon receipt of the READ_DEVICE_TYPE command, the legacy device first issues a LEGACY_ACK byte to its host port. When the Q-PORT device receives this, it adds the header with a logical device address of 0x00 and a byte count of 0x1. The port manager 114 receives this packet and does not pass it on to the device driver indicated in its routing table for device 0x00; instead, it sends the data (which is the LEGACY_ACK byte) to the configuration manager 122 as the auto-configuration cycle is still in progress. The configuration manager 122 knows that the LEGACY_ACK byte is an invalid device type, so it ignores it.

Next, the legacy device issues the real LEGACY_DEVICE_TYPE byte, which the Q-PORT device packetizes and sends via the port controller 110 and port manager 114 to the configuration manager 122. The configuration manager 122 sees that the device driver for this device type is not loaded in memory, so it searches the mass storage device for the corresponding device driver and loads it into memory. The configuration manager 122 then sets up the packet routing table for the port manager 114 to direct packets to and from the device at logical device address 0x00 to the new device driver.

Next, the configuration manager 122 issues a packet containing a READ_DEVICE_TYPE command to the device at logical device address 0x01. The Q-PORT device receives this and responds with a packet containing the device type of 0x80. The configuration manager 122 receives this byte, and sees that it already has this device driver in memory, so all it need do is point the packet routing table in the port manager 114 to this device driver for all packets to or from logical device address 0x01.

As the configuration manager 122 has attached all the relevant device drivers to the Q-PORT system, it sends ENABLE commands to each device and device driver in turn and signals the port manager 114 that the auto-configuration cycle has finished and the configuration manager 122 exits.

Next, the legacy device at logical device address 0x00 is removed. Nothing happens until something is sent to the legacy device. The transmitted packet arrives at the Q-PORT device, which removes the header from the packet and sends the first byte of the data block to the Q-PORT device's slave port. The Q-PORT device detects that its slave port has become disconnected. In response, the Q-PORT device sends a RESET_REQ byte out its host port. The RESET_REQ byte is routed to the configuration manager 122, which initiates an auto-configuration cycle by sending the RESET command out. The RESET response indicates that only one device is in the device chain and it is at logical device address 0x00. The configuration manager 122 queries this device and discovers it is the Q-PORT device with the device type of 0x80. The configuration manager 122 recognizes that the device driver is in memory for the device type

of 0x80. The configuration manager 122 modifies the packet routing table by changing the logical device address associated with the Q-PORT device's device driver. The unused device driver for device type 0x00 is unloaded from memory and the configuration manager 122 signals the end of the auto-configuration cycle to the port manager 114 and exits.

Referring to FIG. 9, several peripherals (for example, a Q-port or "active" peripheral device 904, and a conventional, legacy device 906) may be coupled via a peripheral router 902 to host computer system 900. Peripheral router 902 properly routes signals from host computer system 900 to the device chain represented by devices 904 and 906, and vice versa, in each case in the manner shown in the logical diagram of FIG. 1.

One peripheral router arrangement 1000, shown in FIGS. 10a and 10b, provides three ports: a male mini 6 DIN port 1002 that couples to host computer system 900, and two female mini 6 DIN ports 1004 and 1006 that couple peripheral router 1000 to an active q-port device 904 and a "passive" conventional device 906 respectively. For clarity, FIG. 10b shows the same signal lines "stretched out". When only a passive legacy device such as a conventional keyboard 906 is attached to its respective port 1006, two position, dual-throw switch 1010 is normally "open", so that keyboard data and clock lines properly route through peripheral router 902 to respective PC data and clock lines. When an active q-port device 904 is also attached to its respective port 1004, switch 1010 closes, routing the legacy keyboard data and clock lines to the slave data and clock lines of q-port device 904. In effect, when both a q-port and legacy device are attached, peripheral router 902 creates a linear device chain where pc data and clock signals are first routed to the host data and clock lines of q-port device 904. Then q-port device 904 transports any appropriate signals to and from legacy device 906 via separate slave data and clock lines. Peripheral router 902 provides ground and Vcc to both devices.

Enhanced peripheral router 1102, shown in FIG. 11, provides similar DIN ports 1114 and 1116 for attaching a q-port and a legacy device respectively, but accomplishes the routing of signals from host port 1100 to the appropriate peripheral port via CMOS gates 1118a and 1118b (using a 4066 CMOS gate circuit). One method for detecting when a q-port device is attached to port 1114 is based on placing a low voltage (nominally +5 V) on the male connector (or similar connector) of the q-port device. Legacy device shields are typically coupled to ground or floating. The shield of connector Q-port connector 1114 is coupled via a 100 k resistor 1123 to Vcc and via a much larger 1 Mohm resistor 1125 to ground. When the q-port device is inserted into port 1114, its imposed shielding voltage drives pass-thru line 1120 high, opening gates 1118a and 1118b (which are active-low gates), thereby forcing control and data lines 1122a and 1124a first to the q-port interface 1114, and then from the q-port device to host interface 1100 (via lines 1122b and 1124b). When no q-port device is attached to interface 1114, gates 1118a and 1118b remain open, coupling control and data lines 1122a and 1124a directly to host interface 1100.

Symmetric peripheral router 1202 (FIG. 12) employs similar CMOS gate logic as router 1102, but provides two identical connector ports 1210a and 1210b, into both of which a q-port device or a legacy device can be attached. Gates 1218a and 1218b cut off the control and data connection of connector interface 1210a from host interface 1200 (and route them instead through connector interface 1210b) when a q-port device is inserted into connector

interface 1210b. This cutoff occurs because the low voltage on the shielding of the q-port device transfers to shield line 1222b where it, inverted by inverter 1220b into a null signal, turns off gates 1218a and 1218b via signal line(s) 1224a. The same action occurs when a q-port device is inserted into the first connector interface 1210a: the low voltage on its shielding transfers to shield line 1222a where it, inverted by inverter 1220a into a null signal (transmitted via signal line(s) 1224b), closes gates 1218c and 1218d. Data and clock signals from the other connector interface 1210b are then routed through connector interface 1210a, through its attached q-port device, and then out to host interface 1200. Symmetric peripheral router 1202 thereby allows a q-port device to be plugged into either of the two connector interfaces 1210a or 1210b.

Another symmetric peripheral router 1302, shown in FIG. 13, employs a microcontroller 1330 (Zilog ZB6C03PSC) to couple control and data signals correctly among connector interfaces 1310a and 1310b and host interface 1300, thereby implementing the peripheral attachment scheme. A q-port device can be inserted into either of the two connector interfaces 1310, for example 1310a. In this example, microcontroller 1330 routes all signals first to the q-port-attached connector interface 1310a, along its primary clock and data signal runs 1332a and 1334a. If a signal packet is destined for devices attached to the other connector interface 1310b, such signal packet is passed back through connector interface 1310a along auxiliary clock and data signal runs 1336a and 1338a to microcontroller 1330, which then implements the proper addressing protocols and sends the packet on the primary clock and data signal runs 1332b and 1334b to the other connector interface 1310b. If the q-port device is inserted into the other connector interface 1310b, then these operations and the respective signal wires are reversed.

Multiple device chains can be created from any of the peripheral routers described by inserting the host interface of one router into the connector (or slave) interface of another router, in an unlimited fashion, as shown in FIG. 14. Or, referring to FIG. 15, a microcontrolled peripheral router 1402 can implement a packet routing scheme internally among a plurality of peripheral devices. Microcontroller 1430 receives and transmits data and control packets to and from host computer 900 via host interface port 1400. Microcontroller 1430 then implements the packet routing scheme among a plurality of connector interfaces 1410a through 1410c, as an example. During auto-configuration, each connector interface 1410 having an attached device (whether q-port or legacy) is assigned a unique ascending address. Addressed packets sent from the PC (or from any other attached q-port device) can then be transferred to the appropriate attached peripheral device by stripping its address and transferring the packet to the appropriate connector interface 1410. Data or control packets sent back from an attached peripheral device can have its origination address attached to its packet by microcontroller 1430 for transfer to any other peripheral device and/or computer 900.

Other embodiments are also within the scope of the claims. For example, other types of serial interfaces can be used and the packet and broadcast command widths can be varied.

What is claimed is:

1. Apparatus for coupling two peripheral devices to a single port of a computer, comprising:
 - two peripheral ports for coupling to the peripheral devices;
 - a host port for coupling to the single port of the computer;

15

routing circuitry for selectively coupling one of the peripheral ports to the host port based on the identity of a peripheral device connected to the one peripheral port;

wherein the first peripheral port accepts an active peripheral device; and

wherein the active peripheral device further comprises:
 a port; and
 slave control and data circuitry coupled to the port, where the slave control and data circuitry, upon receiving from the port a slave control and/or data signal intended for another peripheral device, transfers the slave control and/or data signal back to the port.

2. The apparatus of claim 1 wherein the second peripheral port accepts a passive peripheral device.

3. Apparatus for coupling two peripheral devices to a single port of a computer, comprising:
 two peripheral ports for coupling to the peripheral devices;
 a host port for coupling to the single port of the computer;
 routing circuitry for selectively coupling one of the peripheral ports to the host port based on the identity of a peripheral device connected to the one peripheral port;

wherein the routing circuitry couples data and clock signals from the second peripheral port to the first peripheral ports when an active peripheral device is coupled to the first peripheral port; and

wherein the routing circuitry decouples data and clock signals from the second peripheral port to the host port, when said active peripheral device is coupled to the first peripheral port.

4. Apparatus for attaching peripheral devices to a port of a computer, comprising:
 peripheral ports;
 a host port coupled to the port of the computer; and
 routing circuitry selectively coupling the peripheral ports to the host port and to each of the peripheral ports, wherein the routing circuitry can receive a data packet from one of the peripheral ports designated for a peripheral device coupled to a designated one of the plurality of peripheral ports and transmit the data packet to the designated peripheral port.

5. The apparatus of claim 4 wherein the routing circuitry comprises a microprocessor.

6. A method for coupling two peripheral devices, to a single port of a computer, comprising:
 coupling the two peripheral devices to two peripheral ports;
 coupling a host port to the single port of the computer; selectively coupling one of the peripheral ports to the host port based on the identity of the peripheral device connected to the one peripheral port, wherein the first peripheral port accepts an active peripheral device;

wherein the active peripheral device further comprises:
 a port; and
 slave control and data circuitry coupled to the port, where the slave control and data circuitry, upon receiving from the port a slave control and/or data signal intended for another peripheral device, transfers the slave control and/or data signal back to the port.

16

7. The method of claim 6 wherein the second peripheral port accepts a passive peripheral device.

8. A method for coupling two peripheral devices to a single port of a computer, comprising:
 coupling the two peripheral devices to two peripheral ports;
 coupling a host port to the single port of the computer; selectively coupling one of the peripheral ports to the host port based on the identity of the peripheral device connected to the one peripheral port, wherein a routing circuitry couples data and clock signals from the second peripheral port to the first peripheral port, when an active peripheral device is coupled to the first peripheral port; and
 wherein the routing circuitry decouples data and clock signals from the second peripheral port to the host port, when said active peripheral device is coupled to the first peripheral port.

9. Method for attaching peripheral devices to a port of a computer, comprising:
 coupling peripheral ports to the peripheral devices;
 coupling a host port to the port of the computer; and
 selectively coupling the peripheral ports to the host port and to each of the peripheral ports;
 wherein a routing circuitry can receive a data packet from one of the peripheral ports designated for a peripheral device coupled to a designated one of the plurality of peripheral ports and transmit the data packet to the designated peripheral port.

10. The method of claim 9 wherein the routing circuitry comprises a microprocessor.

11. Apparatus for coupling two peripheral devices to a single port of a computer, comprising:
 two peripheral ports for coupling to the peripheral devices;
 a host port for coupling to the single port of the computer; and
 routing circuitry for selectively coupling one of the peripheral ports to the host port based on the identity of a peripheral device connected to the one peripheral port;

wherein the first peripheral port accepts either an active or a passive peripheral device, the active peripheral device further comprises:
 a port; and
 slave control and data circuitry coupled to the port, where the slave control and data circuitry, upon receiving from the port a slave control and/or data signal intended for another peripheral device, transfers the slave control and/or data signal back to the port.

12. The apparatus of claim 11 wherein the routing circuitry selectively couples the one peripheral port to the other peripheral port when an active peripheral device is coupled to the one peripheral port.

13. The apparatus of claim 11 wherein said routing circuitry, upon detecting said active peripheral device coupled to one peripheral port, selectively couples the one peripheral port to the host port, decouples the other peripheral port from the host port and couples the host-decoupled peripheral port to the host-coupled peripheral port.

14. The apparatus of claim 11 wherein the active peripheral device receives control and data signals on a first pair of signal connectors, and transfers back the slave control and data signals on a second pair of signal connectors.

17

15. Apparatus for coupling two peripheral devices to a single port of a computer, comprising:
- two peripheral ports for coupling to the peripheral devices;
 - a host port for coupling to the single port of the computer;
 - routing circuitry for selectively coupling one of the peripheral ports to the host port based on the identity of a peripheral device connected to the one peripheral port;
 - wherein the first and second peripheral ports accepts either an active or a passive peripheral device, the active peripheral device further comprises:
 - a port; and
 - slave control and data circuitry coupled to the port, where the slave control and data circuitry, upon receiving from the port a slave control and/or data signal intended for another peripheral device, transfers the slave control and/or data signal back to the port.
16. Apparatus for coupling two peripheral devices to a single port of a computer, comprising:
- a first peripheral port coupled to an active peripheral device;
 - a second peripheral port coupled to a passive peripheral device;
 - a host port coupled to the computer;
 - routing circuitry coupling the first peripheral port to the host port based on the identity of the active peripheral device coupled to the first peripheral port, the active peripheral device further comprises:
 - a port; and
 - slave control and data circuitry coupled to the port, where the slave control and data circuitry, upon receiving from the port a slave control and/or data signal intended for another peripheral device, transfers the slave control and/or data signal back to the port.
17. A method for coupling two peripheral devices to a single port of a computer, comprising:
- coupling the two peripheral devices to two peripheral ports;
 - coupling a host port to the single port of the computer, selectively coupling one of the peripheral ports to the host port based on the identity of the peripheral device connected to the one peripheral port;
 - wherein the first peripheral port accepts either an active or a passive peripheral device, and wherein the active peripheral device further comprises:

18

- a port; and
 - slave control and data circuitry coupled to the port, where the slave control and data circuitry, upon receiving from the port a slave control and/or data signal intended for another peripheral device, transfers the slave control and/or data signal back to the port.
18. The method of claim 17 wherein the active peripheral device receives control and data signals on a first pair of signal connectors, and transfers back the slave control and data signals on a second pair of signal connectors.
19. A method for coupling two peripheral devices to a single port of a computer, comprising:
- coupling the two peripheral devices to two peripheral ports;
 - coupling a host port to the single port of the computer;
 - selectively coupling one of the peripheral ports to the host port based on the identity of the peripheral device connected to the one peripheral port,
 - wherein the first and second peripheral ports accepts either an active or a passive peripheral device, the active peripheral device further comprises:
 - a port; and
 - slave control and data circuitry coupled to the port, where the slave control and data circuitry, upon receiving from the port a slave control and/or data signal intended for another peripheral device, transfers the slave control and/or data signal back to the port.
20. A method for coupling two peripheral devices to a single port of a computer, comprising:
- coupling a first peripheral port to an active peripheral device;
 - coupling a second peripheral port to a passive peripheral device;
 - coupling the first peripheral port to the host port based on the identity of the active peripheral device coupled to the first peripheral port;
 - wherein the active peripheral device further comprises:
 - a port; and
 - slave control and data circuitry coupled to the port, where the slave control and data circuitry, upon receiving from the port a slave control and/or data signal intended for another peripheral device, transfers the slave control and/or data signal back to the port.

* * * * *